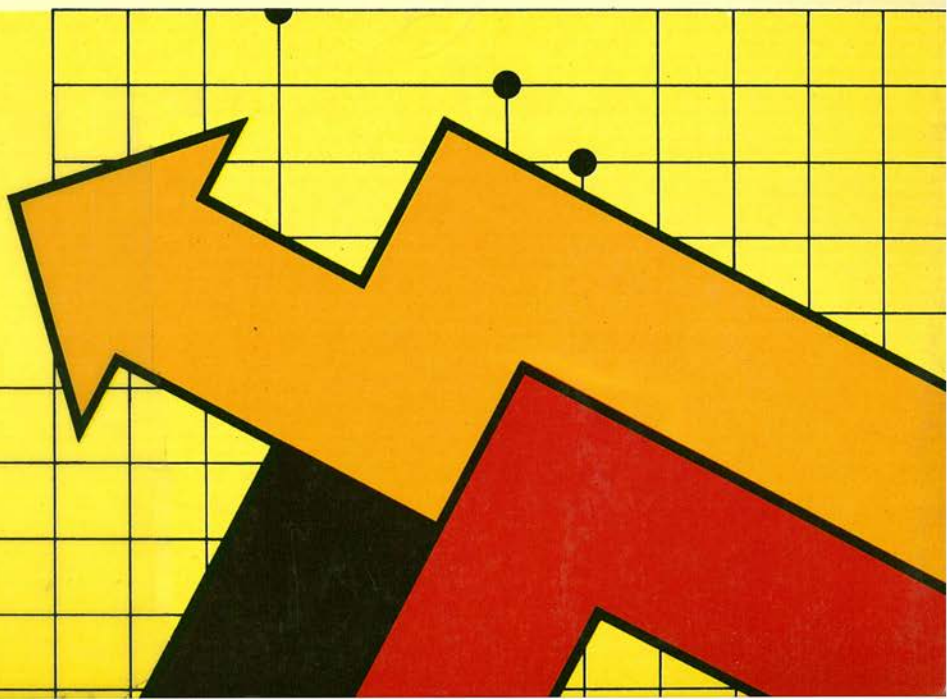
INTERFACCIAMENTO DI MICROCOMPUTER

ESPERIMENTI UTILIZZANTI IL

CHIP 8255 PPI

EDIZIONE ITALIANA

PAUL F. GOLDSBROUGH
con esperimenti di
PETER R. RONY

GRUPPO EDITORIALE JACKSON

INTERFACCIAMENTO DI MICROCOMPUTER

ESPERIMENTI UTILIZZANTI IL

CHIP 8255 PPI

di
Paul F. Goldsbrough
con esperimenti di
Peter R. Rony

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

INTRODUZIONE ALL'EDIZIONE ITALIANA

Quando, nella fase pre-progettuale di un sistema, ci si trova davanti al problema della scelta del microprocessore, le caratteristiche dei singoli microprocessori non sono elementi sufficienti per giustificare una scelta al posto di un'altra. Il microprocessore va infatti visto come componente di una famiglia, ci si consenta l'analogia, non va vista solo nei confronti del "padre", ma dal complesso delle bontà dei singoli componenti. La conoscenza quindi, degli altri componenti la famiglia, diviene elemento essenziale per motivare la scelta di un microprocessore.

L'8255 (interfaccia periferica programmabile) è un componente della famiglia 8080. Esso come LSI programmabile, riveste un ruolo molto importante come responsabile dell'I/O parallelo tra la CPU, la memoria ed il mondo esterno.

Il volume (tutto sull'8255), non va considerato alla stregua di una application note. Come l'Autore fa infatti notare nella Sua prefazione, il chip è il punto di partenza per scrivere un libro a carattere generale sugli LSI programmabili.

Gli esperimenti vengono realizzati attraverso la tecnica del breadboarding usando una basetta SK-10 ed utilizzando come microcomputer un Mini-Micro-Designer (MMD-1). Questi strumenti didattici prodotti dalla E & L Instrument come gli altri riportati nel testo, sono distribuiti in Italia dalla Microlem di Milano. È chiaro da ciò l'intento dell'Autore di dare un taglio sperimentale al libro, coerentemente agli altri volumi della serie *Circuiti logici e di memoria Vol. 1 e Vol. 2,* (già Bugbook I e II); *Interfacciamento e programmazione del microcomputer 8080* (Bugbook III); *Esperimenti con TTL e 8080A Vol. 1 e Vol. 2,* (già Bugbook V e VI); *L'interfacciamento fra microcomputer e convertitori analogici* (Bugbook VII).

La Scuola di Elettronica

PREFAZIONE

Questo libro fu cominciato all'inizio del 1977, quando i due volumi (già Bugbook V e Bugbook VI) *Esperimenti con TTL e 8080A Vol. 1 e Vol. 2,* erano nella fase finale di preparazione. Io avevo appena terminato uno studio di sei mesi al College of Advanced Education di Camberra ed ero venuto a Blacksburg per lavorare con gli autori delle Blacksburg Continuing Education Series, per un periodo di tre mesi. Durante la mia permanenza con gli autori feci alcuni esperimenti preliminari con il circuito integrato 8255 Interfaccia Programmabile Periferica (PPI) della Intel Corporation, che furono scritti. Dopo aver completato gli esperimenti e letto le informazioni disponibili circa l'8255, fui stimolato dal concetto di circuito integrato software-programmabile generalizzato, così proposi di scrivere un libro sul chip 8255 ed espandere e rivedere gli esperimenti per fare un libro general-purpose, che potesse piacere a studenti, sperimentatori, a chi ha l'hobby del computer, a ingegneri, scienziati e altri utilizzatori di chip d'interfaccia programmabili.

Negli stadi iniziali della preparazione, emersero due importanti punti:

1. I vari modi di operare del PPI, riflettono la maggior parte delle tecniche di ingresso/uscita parallelo di dati, usate con microcomputer.
2. L'impiego di circuiti integrati generalizzati LSI software-configurabili, che concentrano hardware d'interfaccia per microcomputer, aumenterà. La flessibilità è ottenuta con un progetto che permette di cambiare la configurazione dei dispositivi, tramite l'invio di *byte di controllo* a un *registro di controllo* interno ai circuiti integrati.

Da qui nacque la linea del libro così che:

- Le principali tecniche d'ingresso/uscita per microcomputer, sono introdotte dal Capitolo 3 fino al 7 e la loro implementazione è illustrata attraverso l'impiego dell'8255 PPI.
- Le procedure per la configurazione del PPI sono introdotte in maniera generalizzata nel Capitolo 2, cosicchè, una volta approfondito il procedimento, possono essere applicate ad altre interfacce software-configurabili (programmabili) con difficoltà minima. Mentre i particolari delle loro applicazioni cambieranno, il procedimento per il loro impiego, tramite l'accesso ai loro registri di controllo e dati (e possibili altri), rimarrà sostanzialmente lo stesso.

Si suppone che il lettore abbia approfondito le fasi di programmazione ed interfacciamento di microcomputer. Ciò comprende gli argomenti riguardanti la generazione d'impulsi di selezione di dispositivi, interrupt da polling o con vettori, I/O in mappa di memoria e accumulatore, così come programmazione in linguaggio assembler. Gli argomenti riguardanti I/O in mappa di memoria e accumulatore, sono stati rianalizzati nel Capitolo 1, a causa della loro importanza nei collegamenti di interfaccia, fra il chip microprocessore 8080A o il microcomputer basato sull'8080A e il chip 8255. Per i particolari sull'interfacciamento di microcomputer, il lettore è rimandato ai due volumi citati in precedenza. Per i dettagli sulla programmazione di microcomputer, rimandiamo ai due volumi *Progetto Software 8080/8085.*

Gli esperimenti sono stati progettati per rafforzare i concetti di ogni Capitolo. Gli esperimenti presuppongono una qualche familiarità con l'elettronica digitale e l'assemblaggio di piastre. Per l'esecuzione degli esperimenti sarà necessaria un po' di circuiteria aggiuntiva. Di solito dispositivi digitali ausiliari come monitor, generatori d'impulsi, switch logici e clock, vengono tutti quanti usati negli esperimenti. Vengono anche richieste alcune serie di circuiti integrati SN7400.

Così come altri libri della Blacksburg Continuing Education Series, questo è stato concepito per essere autoistruttivo e, a questo scopo, vengono fornite le risposte a tutti i problemi degli esperimenti. Dal momento che probabilmente le vostre risposte concorderanno con le nostre osservazioni, si può tentare di andare oltre facilmente senza un'analisi del *perchè* furono fatte le varie osservazioni e a cosa erano collegate.

Nei corsi base di laboratorio, dove il tempo è limitato, c'è la tendenza a soccombere a questa tentazione, senza essere consapevoli di ciò che deve avere spazio in un circuito sperimentale. Attenzione, questi esperimenti possono fornire un'esperienza molto remunerativa. Si è scoperto anche,

che molti dei chip d'interfaccia più complessi impiegano tecniche che sono simili a quelle usate per l'8255. La comprensione del chip PPI, vi permetterà così un buon inizio quando sarete interessati all'uso di alcuni degli ultimi chip d'interfaccia.

Il materiale presentato in questo libro è stato controllato con successo in entrambe le situazioni di corso di laboratorio formale e in un breve corso di presentazioni in cui si sono combinate conferenze con esperienze pratiche. I lettori in Australia, possono contattarmi direttamente per informazioni addizionali.

Questo libro non sarebbe potuto essere completato senza l'aiuto di molte persone. Sono in particolar modo debitore di Peter Rony, Dave Larsen e Jon e Chris Titus per i loro utili consigli e incoraggiamenti durante la preparazione di questo libro. Dal Canberra CAE, i miei ringraziamenti a Roberta Vetter e a Margaret Bonnet per le molte ore passate a scrivere con cura le bozze e poi il manoscritto finale; a Tony Howkins e John Houldsworth, per la preparazione dei diagrammi; a Steve Morland il buon lavoro nel Capitolo 7 e la preparazione dei programmi finali assemblati.

Infine, il mio amore e ringraziamento a mia moglie, Anne, il cui sostegno, incoraggiamento, pazienza e comprensione durante la scrittura, fu così necessario e così spontaneamente vicino. Dedico a lei questo libro.

PAUL F. GOLDSBROUGH

Sommario

CAPITOLO 6

CAPITOLO 7

APPENDICE 1

APPENDICE 2

CAPITOLO 1

INTRODUZIONE AL CHIP 8255 PPI

1-1. ALCUNE DOMANDE E RISPOSTE

Nei libri e nelle riviste che trattano di microcomputer, si sta facendo continuamente riferimento a Interfacce Periferiche Programmabili (Programmable Peripheral Interface), o PPI. Tale dispositivo viene largamente impiegato dai costruttori di apparecchiature OEM (Original Equipment Manufacturers, OEM), come circuito integrato ingresso/uscita, per prodotti basati su microcomputer. Esso viene usato sempre più frequentemente anche da ingegneri elettronici, tecnici e praticanti hobby.

- Che cos'è?
- Dove conviene come elemento in un sistema a microcomputer?
- Perchè usarlo?
- Come si usa?

Questi sono i problemi che discuteremo nelle pagine seguenti, sfruttando come esempio l'Interfaccia Periferica Programmabile 8255 della Intel.

Il PPI è un chip a 40 pin circuito-integrato larga-scala (LSI), che viene impiegato nei microcomputer come *interfaccia* fra il bus dati del microcomputer e dispositivi esterni ingresso/uscita. Tale descrizione generale può essere applicata anche ad altri chip ingresso/uscita, per esempio, l'interfaccia programmabile per comunicazioni (8251), o USART. Il PPI, differisce dall'USART per il fatto che esso è progettato per il *trasferimento parallelo di dati*, mentre l'USART è sfruttato per la trasmissione seriale.

Mentre i dettagli della loro applicazione in un sistema a microcomputer sono certamente diversi, l'approccio di base per il loro impiego è lo stesso. Per questo motivo, quando si sarà completato questo libro e si saranno approfondite le tecniche di base richieste per l'impiego del PPI,

si troverà molto più facile comprendere e utilizzare l'USART e altri dispositivi d'ingresso/uscita programmabili. (Tali dispositivi comprendono l'Adattatore d'Interfaccia Programmabile 6820 (PIA) e l'Adattatore d'Interfaccia Asincrono 6850 (ACIA) della Motorola, l'UART 6011 TMS della Texas Instruments ed il Controllore d'Interrupt 8259 della Intel).

La Fig. 1-1, mostra uno schema a blocchi di un microcomputer tipico basato sull'8080 della Intel. Tale schema illustra la posizione del PPI e il suo legame con gli altri componenti del sistema. Per chiarezza non sono state rappresentate tutte le linee dati, indirizzi e di controllo. Si noti che il PPI è collegato al microcomputer attraverso il bus dati e che esso collega il microcomputer al mondo esterno per mezzo di 24 linee I/O. Queste sono in genere suddivise in tre byte di 8-bit, che sono indicati con PA0-PA7, PB0-PB7 e PC0-PC7.

Ci sono due grandi vantaggi nell'impiego del PPI in un sistema a microcomputer. Il primo è che il PPI concentra operazioni d'ingresso/uscita parallelo in un solo circuito integrato, a meno che non siano richieste più di 24 linee ingresso/uscita. Dal momento che tutta la logica ingresso/uscita è su un circuito integrato, sia la complessità d'interfacciamento che il numero di chip, sono ridotte, con il risultato di una diminuzione del costo. Il secondo e forse il più importante vantaggio nell'impiego del PPI, è la grande flessibilità che esso offre per l'interfacciamento I/O di microcomputer. Tale flessibilità è ottenuta rendendo il PPI *software configurabile* - da qui il suo nome, l'interfaccia periferica *programmabile*. La configurazione delle sue 24 linee I/O come ingressi, uscite o forse I/O bidirezionale, è quindi sotto il controllo del *software* anzichè dell'hardware. Questo rende molto più facile l'assegnazione delle linee di I/O ed ogni successiva modifica. Se ad esempio, si desidera aggiungere un dispositivo extra di ingresso o uscita al proprio sistema a microcomputer, oppure si decide di cambiare il tipo di periferica che si sta usando passando da un display luminoso a un terminale video, tutto quello che è necessario, a parte forse, in alcuni casi, l'aggiungere o il togliere una o due linee di controllo, è cambiare il proprio software di ingresso/uscita.

È chiaro il concetto? L'autore pensa di sì e in pratica è semplice, una volta analizzata l'elettronica del chip e le procedure d'uso. Questo è il compito dei prossimi capitoli. Innanzi tutto cominceremo, comunque, con un'analisi delle tecniche base d'ingresso/uscita. Prima di far ciò è certamente giusto in questa introduzione del PPI, dichiarare alcuni dei suoi difetti.

Il concentrare le linee di I/O in un'unica area, può non essere sempre desiderabile. Il software per la configurazione del dispositivo allunga anche, in media, di circa 10 byte il programma e ciò può portarlo a non

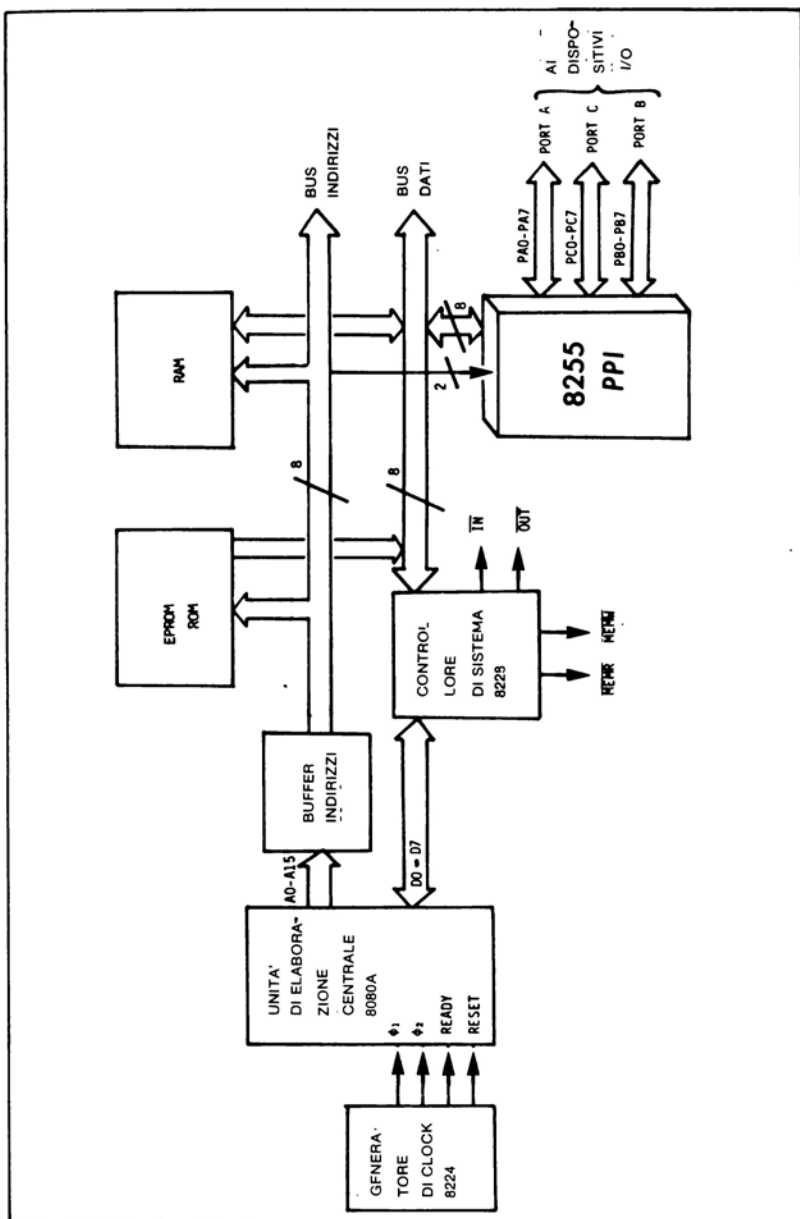



Fig. 1-1. Sistema generalizzato basato sul microcomputer 8080A, che mostra la posizione funzionale dell'Interfaccia Periferica Programmabile.

essere più contenuto in 1K byte, con la conseguenza di aver bisogno di una seconda ROM per la memorizzazione del programma. Oltre a ciò, una volta stabilite le specifiche per l'interfaccia ingresso/uscita, può essere meno dispendioso impiegare porte I/O standard del tipo di quelle che saranno descritte nel paragrafo successivo, assegnando perfino costi aggiuntivi per lo schema della piastra del circuito stampato. Infine, le uscite del PPI (8255) non hanno il fan-out completo dei circuiti integrati della serie 7400.

1-2. OPERAZIONI INGRESSO/USCITA DI BASE

Il compito principale di trasferimenti I/O è: sia trasferire dati digitali da un registro interno al microcomputer ad un dispositivo esterno d'uscita, come ad esempio una telescrivente, una tastiera, altri computer, relè e così via; sia ricevere in ingresso dati digitali da un dispositivo esterno al microcomputer. I dati digitali possono essere trasferiti in maniera *seriale*, nel qual caso si può usare un'UART; o possono essere trasferiti in maniera *parallela*. Qui tratteremo solamente quest'ultima classe di trasferimenti I/O.

La trasmissione parallela di dati digitali o byte di informazione al e dal microcomputer, può essere compiuta in entrambi i modi, ognuno dei quali è associato ai registri interni del microcomputer. La Fig. 1-2 mostra i sette registri disponibili sull'8080A. Questi comprendono sei registri general-purpose (B, C, D, E, H e L) e l'accumulatore. Quest'ultimo è un registro speciale, tutte le operazioni aritmetiche e logiche vengono eseguite con un dato di un byte che è conservato nell'accumulatore. In generale, quando i byte di dati vengono inseriti o estratti dall'accumulatore, la procedura d'ingresso/uscita è conosciuta come un *I/O in accumulatore*. Se il byte dato è trasferito direttamente da un dispositivo I/O esterno a un registro general-purpose, la tecnica è conosciuta come un *I/O in mappa di memoria*.

(A) I/O in accumulatore

L'I/O in accumulatore è la più semplice tecnica ingresso/uscita da comprendere e usare. Servono solamente due istruzioni speciali per trasferire i dati dall'accumulatore a un dispositivo esterno. Al dispositivo esterno viene assegnato uno dei 256 codici che sono forniti dal codice di un dispositivo ad 8-bit. Per inserire i dati si usa l'istruzione

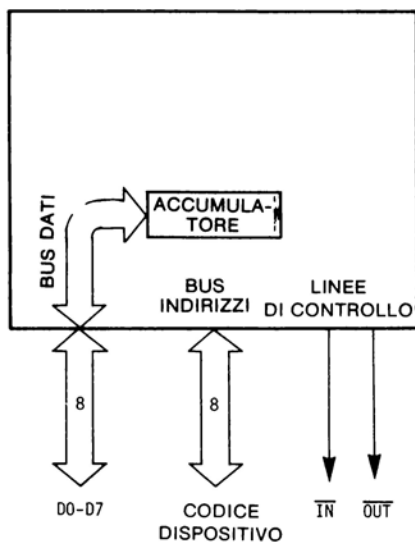
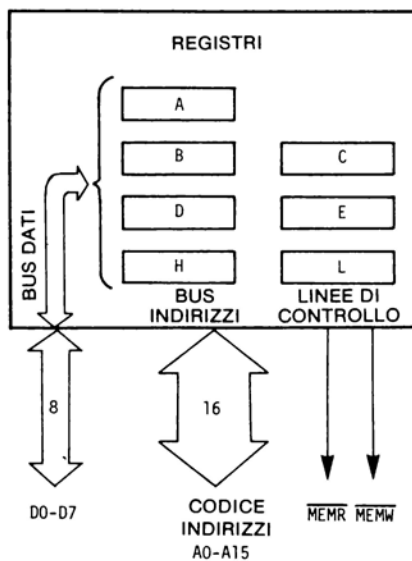
IN
< B2 >

(A) I/O in Accumulatore

(B) I/O in Mappa di Memoria

Fig. 1-2. Rappresentazione schematica di I/O in accumulatore ed in mappa di memoria.

dove < B2 > è il codice dispositivo a 8-bit del dispositivo di ingresso. Durante l'esecuzione di tale istruzione, un segnale di controllo, $\overline{IN}$, è portato al valore logico basso dal chip 8228 controllore del sistema (*cf* Fig. 1-1), quando il codice < B2 > del dispositivo appare sulle linee indirizzi A0-A7. Nei sistemi 8080, il codice del dispositivo è riportato anche sulle linee indirizzi A8-A15. L'impulso IN ed il codice del dispositivo devono poi essere usati per generare un impulso di sincronizzazione per il trasferimento dati, che è conosciuto come *impulso di selezione chip*. Un modo per fare quanto sopra è illustrato in Fig. 1-3. L'impulso di selezione del dispositivo, viene poi usato per abilitare buffer three-state, i quali metteranno sul bus dati, i dati in ingresso da caricare nell'accumulatore all'interno del microprocessore 8080.

L'uscita dei dati con l'I/O in accumulatore, è fatta in maniera analoga. Si usa l'istruzione a due byte

OUT
< B2 >

dove < B2 > è un byte dati di 8 bit, che rappresenta il codice del dispositivo a cui devono essere inviati i dati in uscita. A quell'istante, insieme al codice del dispositivo, il controllore del sistema 8228 genera un impulso $\overline{OUT}$ (impiegando l'informazione di stato dell'8080A). Per generare un impulso di selezione dispositivo, viene ancora usato un circuito come quello mostrato in Fig. 1-3, impiegandolo questa volta per

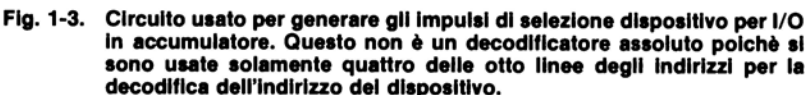
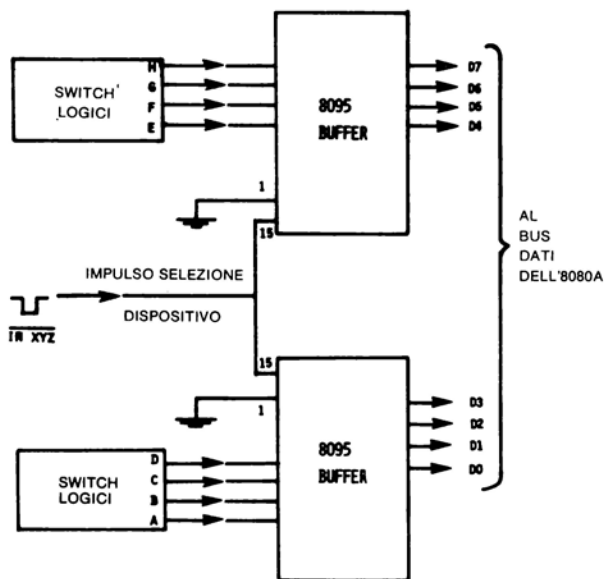
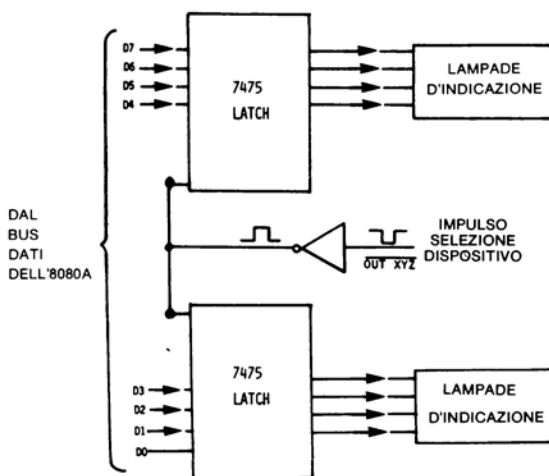
Fig. 1-3. Circuito usato per generare gli impulsi di selezione dispositivo per I/O in accumulatore. Questo non è un decodificatore assoluto poichè si sono usate solamente quattro delle otto linee degli indirizzi per la decodifica dell'indirizzo del dispositivo.

il latching del byte dati che si trova sul bus dati e proveniente dall'accumulatore. Le Fig. 1-4A e 1-4B mostrano, rispettivamente, circuiti per ingresso ed uscita dall'accumulatore.

(A) Per ingresso in accumulatore

(B) Per uscita dall'accumulatore

Fig. 1-4. Circuiti tipici per il buffer dei dati in ingresso ed il latch dei dati in uscita con I/O in accumulatore.

I vantaggi dell'I/O in accumulatore, sono la semplicità e la linearità dell'approccio. L'ingresso o l'uscita di informazioni richiede solamente l'impiego di due istruzioni speciali. Questo è particolarmente vantaggioso quando si usa per la prima volta un microcomputer. Ci sono due possibili inconvenienti in tale tecnica. Il primo e più importante, è che tutti i byte di dati in ingresso e in uscita devono passare attraverso l'accumulatore. In molti casi il luogo d'origine dei dati in uscita, o la destinazione di quelli in ingresso, è la memoria di lettura/scrittura o uno dei registri general-purpose. Ciò significa che saranno richieste istruzioni extra per trasferire i byte di dati al, o, dall'accumulatore. Il tempo extra e le esigenze di memoria possono, in alcuni casi, essere critici. Il secondo inconveniente è che sono disponibili solamente 256 codici per i dispositivi ingresso/uscita. L'autore, sinceramente, dubita che questo secondo aspetto sia realmente un serio problema per la maggior parte degli utilizzatori di microprocessori.

(B) I/O in mappa di memoria

La tecnica usata nell'I/O in mappa di memoria, è un artificio che permette di pensare al dispositivo I/O come a una parte di memoria. Dal momento che il microprocessore non ha realmente modo di sapere a quale tipo di dispositivo di "memoria" sta accedendo, tutto ciò che viene richiesto è:

- Usare istruzioni di trasferimento dati di memoria nelle subroutine di ingresso/uscita.
  Esempi di tali istruzioni sono:

```
MOV r, M      ADC M      CMP M
MOV M, r      SBB M      MVI M
```

  dove
  r rappresenta uno dei sette registri dell'8080A,
  M è la locazione di memoria indirizzata indirettamente dal contenuto della coppia di registri H e L.

- Decodificare gli *impulsi selezione indirizzi*, impiegando le linee indirizzi ed i segnali di controllo read/write di memoria, $\overline{\text{MEMR}}$ e $\overline{\text{MEMW}}$, per selezionare od abilitare un buffer three-state (per l'ingresso) o un latch (per l'uscita).

Procedendo in questo modo, si rendono disponibili 64K di codici di dispositivo (1K = 1024). In pratica, almeno i primi 32K indirizzi (il bit d'indirizzo A15 al valore logico 0), sono riservati alla memoria. Se le necessità di memoria sono modeste, l'altra metà di essa, cioè, gli indirizzi superiori a 32K, vengono assegnati alla memoria I/O (il bit d'indirizzo A15 al valore logico 1). Quanto detto è illustrato in Fig. 1-5.

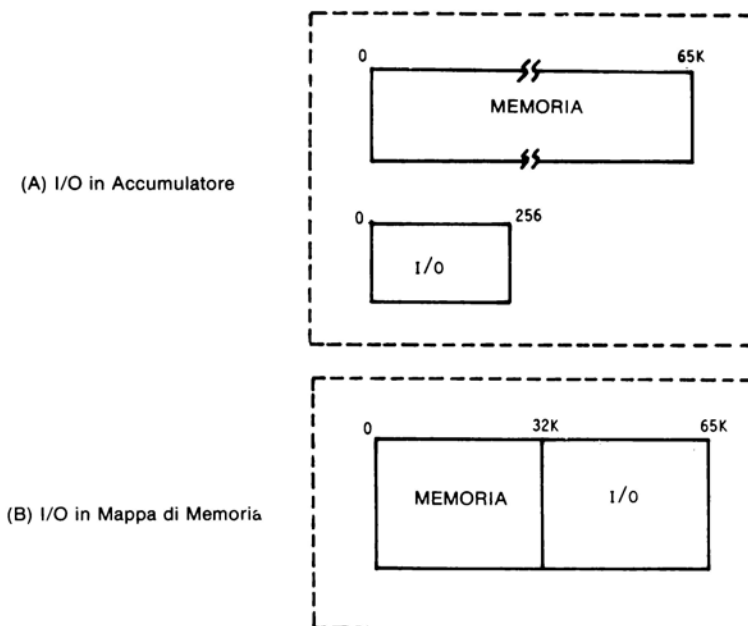



Fig. 1-5. Confronto fra i blocchi di memoria per I/O in accumulatore ed in mappa di memoria.

Solitamente non è richiesta una decodifica *assoluta* o unica di indirizzi di memoria a 16-bit, può quindi essere usato un circuito come quello riportato in Fig. 1-6, il quale sfrutta una decodifica a linea-singola o *in linea retta.* Se sono necessari più di 32K di memoria, l'assegnazione degli indirizzi di I/O può essere ridotta a 16K, 8K, o, 4K, ecc., collegando insieme rispettivamente i bit d'indirizzo A15, A14, ed A13 al posto dell'inverter indicato in Fig. 1-6.

Il concetto di I/O in mappa di memoria, è importante da comprendere, dal momento che, generalmente, i microprocessori non hanno le speciali istruzioni di ingresso e uscita descritte precedentemente nella tecnica di I/O in accumulatore. Il vantaggio dell'I/O in mappa di memoria è che i registri general-purpose possono essere usati per trasferire dati a, e, da periferiche. Possono anche essere eseguite operazioni logiche, direttamente fra un byte nell'accumulatore e il contenuto di una porta d'ingresso (rappresentante di solito l'informazione dello stato di una periferica). In alcuni casi ciò può portare ad un aumento della velocità globale del sistema. Ad ogni modo, l'I/O in mappa di memoria, è molto meno efficiente dell'I/O in accumulatore, in termini di impegno di memoria, dal momento che vincola la coppia di registri H e L (in un

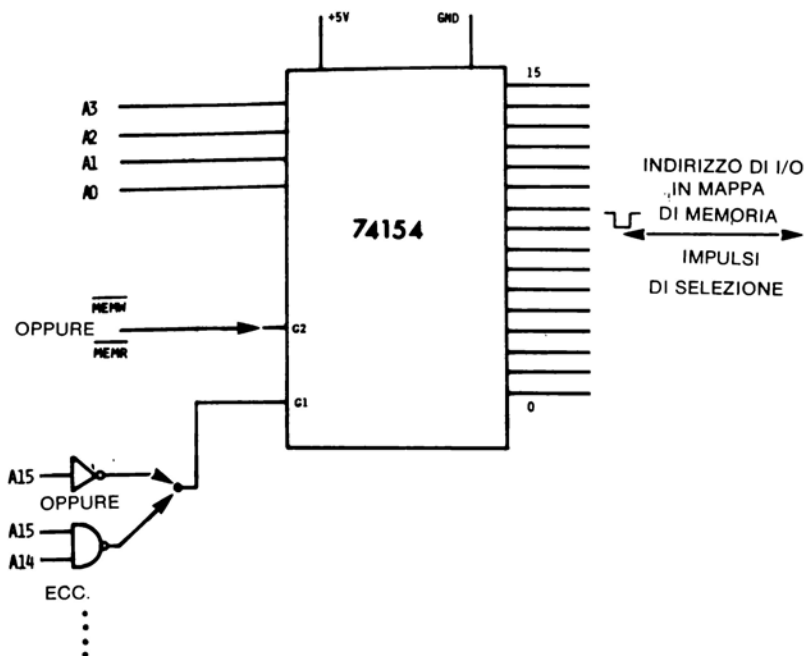



Fig. 1-6. Circuito di decodifica per la generazione d'impulsi di selezione per 16 diversi indirizzi con I/O in memoria. Con l'inverter e l'uso di A15 l'HI byte d'indirizzo è 200 ed il campo del LO indirizzo va da 000 a 017. Si noti che gli indirizzi non sono nè unici nè assoluti.

microcomputer basato sull'8080) per l'indirizzamento, più un altro registro per i dati. Inoltre, l'I/O in mappa di memoria, richiede schemi di decodifica più complessi di quelli dell'I/O in accumulatore.
L'autore ha usato entrambe le tecniche e preferisce l'I/O in accumulatore per ogni caso, ma impone maggior tempo-critico di I/O. Nel libro sono inclusi esempi d'impiego di entrambe le tecniche per interfacciare il PPI all'8080.

Per completare questa breve discussione sulle tecniche di base ingresso/uscita, le tabelle 1-1 e 1-2, riassumono le caratteristiche importanti dell'I/O in accumulatore ed in mappa di memoria. NOTA: sia per l'I/O in mappa di memoria che in accumulatore, assicurarsi sempre che la decodifica degli indirizzi di memoria o il codice dispositivo, siano sufficienti a garantire indirizzi unici per i dispositivi ingresso/uscita. L'estensione di tale decodifica dipende semplicemente dalle dimensioni del sistema microcomputer. In piccoli sistemi, sarà probabilmente adeguato l'impiego di linee indirizzo individuali (decodifica in linea retta). Sistemi più grandi, o sistemi che dovranno probabilmente essere espansi, richiederanno decodifica aggiuntiva.

Tabella 1-1. Sommario delle caratteristiche dell'I/O in accumulatore

Istruzioni 8080A:	OUT< B2 > IN< B2 >
Segnali di controllo:	$\overline{OUT}$ $\overline{IN}$
Trasferimento dati:	Fra accumulatore e dispositivo I/O
Decodifica dispositivo:	Codice dispositivo ad 8 bit, da A0 ad A7 **o** da A8 ad A15, che è il byte< B2 > nell'istruzione IN o OUT.
Terminologia:	I procedimenti I/O sono chiamati **ingresso** ed **uscita**. Il segnale decodificato che funge da strobe per un dispositivo I/O sarà chiamato **impulso selezione dispositivo**.

Tabella 1-2. Sommario delle caratteristiche dell'I/O in mappa di memoria

Istruzioni 8080A:	MOV B,M	MOV M,H	ANA M
	MOV C,M	MOV M,L	XRA M
	MOV D,M	MOV M,A	ORA M
	MOV E,M	STAX B	CMP M
	MOV H,M	STAX D	INR M
	MOV L,M	LDAX B	DCR M
	MOV A,M	LDAX D	MVI M
	MOV M,B	ADD M	STA <B2><B3>
	MOV M,C	ADC M	LDA <B2><B3>
	MOV M,D	SUB M	SHLD<B2><B3>
	MOV M,E	SBB M	LHLD<B2><B3>
Segnali di controllo:	$\overline{MEMR}$ $\overline{MEMW}$		
Trasferimento dati:	Fra dispositivo I/O in mappa di memoria ed i registri B, C, D, E, H, L o l'accumulatore (registro A)		
Decodifica dispositivo:	Codice dispositivo a 16 bit, da A0 ad A15, che è contenuto nella coppia di registri H, B, D oppure nei byte < B2 > e < B3 > delle istruzioni STA, LDA, SHLD, LHLD. In alcuni casi è utile e conveniente riservare l'area di memoria relativa ai 32K superiori per gli indirizzi di I/O in mappa di memoria; si ha I/O in memoria quando A15 del bus degli indirizzi è al valore logico 1. I bit da A0 ad A7 possono essere usati per decodificare uno specifico dispositivo I/O quando A15 = 1. Il dispositivo I/O è fatto in modo da apparire come una locazione di memoria di 8 bit e per leggere da o scrivere nella specifica memoria del dispositivo I/O si impiegano le istruzioni normalmente riferite alla memoria.		
Terminologia:	Le operazioni di I/O in mappa di memoria, saranno chiamate **lettura** e **scrittura**, invece che ingresso ed uscita. Il segnale decodificato che funge da strobe ad un dispositivo I/O in mappa di memoria, sarà chiamato **impulso di selezione indirizzo** invece che impulso di selezione dispositivo.		

1-3. SOMMARIO DEGLI ESPERIMENTI DA 1-1 A 1-5

Gli obiettivi degli esperimenti sono i seguenti:
Esperimenti 1-1 e 1-2: questi esperimenti mostreranno come costruire un semplice bus monitor, un circuito per conteggi e un circuito hardware single-step. I circuiti verranno usati in esperimenti più avanti, e vengono forniti per permettervi di scoprire se il vostro microcomputer basato sull'8080A, li ha già oppure no.
Esperimenti 1-3 fino 1-5: in questi esperimenti vengono illustrate le tecniche di I/O in mappa di memoria e in accumulatore. Se non avete familiarità con questi argomenti dovrete fare questi esperimenti.

Esperimento	*Descrizione*
1-1	Lo scopo di questo esperimento, è di installare una coppia di circuiti su un SK-10, per (a) contare tipi diversi di impulsi di sincronizzazione e (b) mettere su monitor le informazioni presenti sul bus dati bidirezionale.
1-2	Lo scopo di questo esercizio è di costruire un circuito single-step per un microcomputer basato sull'8080A.
1-3	In questo esperimento installerete ed esaminerete il funzionamento del chip circuito-integrato 8212, come porta d'ingresso in mappa di memoria.
1-4	Lo scopo di questo esperimento è di mostrare l'esecuzione di un'operazione AND fra una porta di ingresso in mappa di memoria e l'accumulatore.
1-5	Questo esperimento mostra e confronta il comportamento di due tecniche di I/O, cioè, I/O in mappa di memoria ed in accumulatore, per l'ingresso di dati in un microcomputer 8080.

ESPERIMENTO 1-1
MONITOR DEL BUS E CIRCUITI DI CONTEGGIO

Scopo

Lo scopo di questo esperimento è quello di collegare una coppia di circuiti su di una piastra che (a) conti diversi tipi di impulsi di sincronizzazione e (b) metta su monitor le informazioni presenti sul bus dati bidirezionale.

Configurazione dei pin dei circuiti integrati (Fig. 1-7)

PIN	
1	Catodo Comune
2	Segmento F
3	Segmento G
4	Segmento E
5	Segmento D
6	Catodo Comune
7	Punto decimale
8	Segmento C
9	Segmento B
10	Segmento A

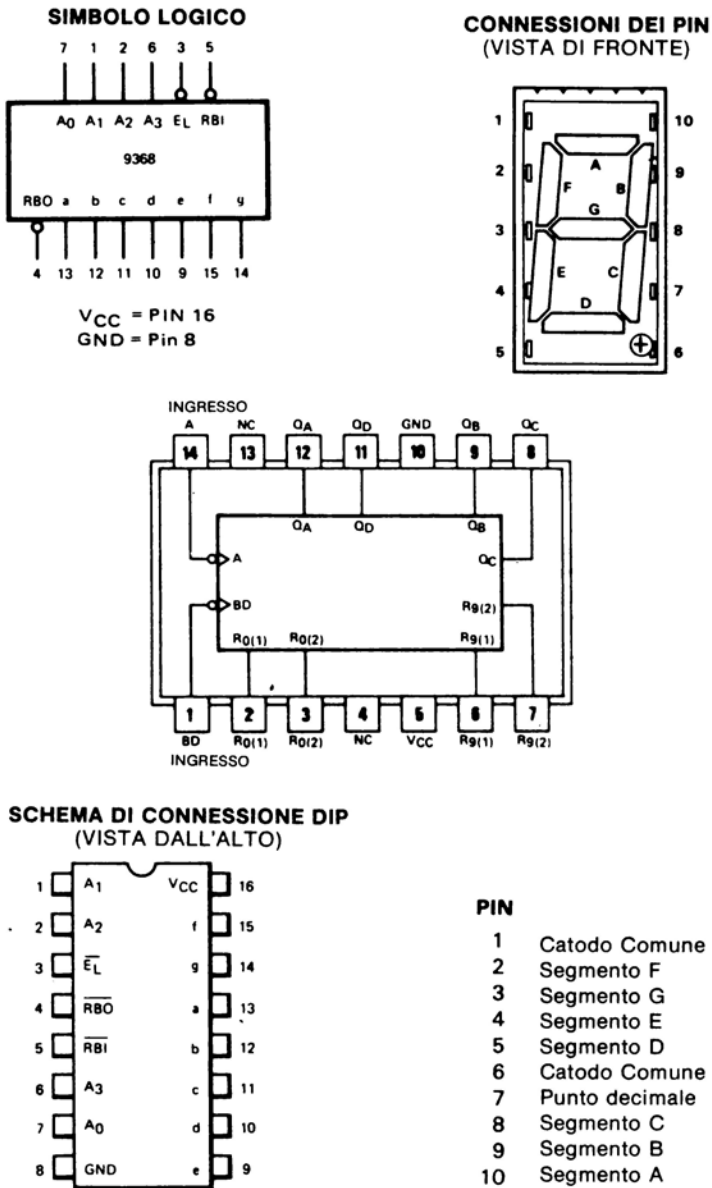



Fig. 1-7. Configurazione dei Pin dell'IC.

Schemi dei circuiti (Fig. 1-8)

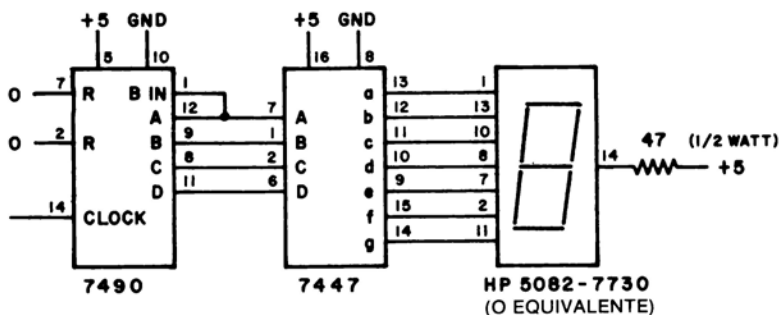
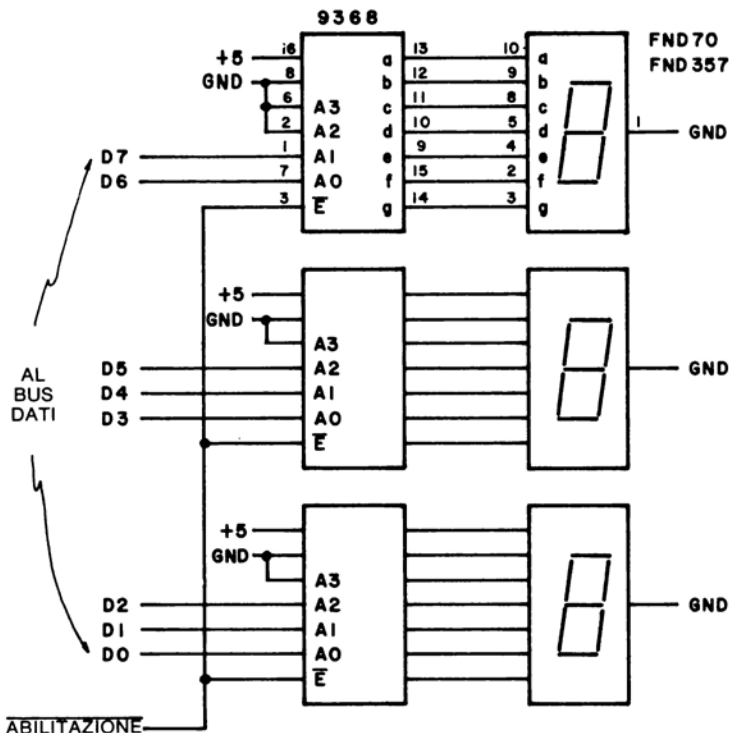
Circuito di Conteggio - In questa configurazione, si è fatto uso di un display su circuito integrato a sette-segmenti ad anodo-comune della Hewlett-Packard (Fig. 1-8A).

(A) Circuito di conteggio

(B) Monitor del Bus

Fig. 1-8. Schemi dei Circuiti.

Monitor del bus - Si è impiegato un circuito integrato latch/decoder/driver 9368 della Fairchild, per pilotare i chip display a sette-segmenti a catodo-comune FND70 (Fig. 1-8B). Per maggior chiarezza sono stati indicati i pin di uscita di una sezione.

Passo 1

Collegare il circuito di conteggio e uno dei circuiti di monitor del bus, preferibilmente su un'unica piastra. Si può voler lasciare spazio per ulteriori chip e/o display per possibili espansioni del circuito.

Passo 2

Elencare, nello spazio sottostante, i tipi di segnali di sincronizzazione che si possono contare con il circuito di conteggio riportato in Fig. 1-8A.

Si possono contare segnali come $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IN}$, $\overline{OUT}$, $\overline{INT}$ e altri segnali di controllo del bus.

Passo 3

Spiegare, nello spazio sottostante, i diversi tipi di ingressi che possono essere applicati ai latch d'ingresso dei circuiti di Fig. 1-8B. Chiameremo ciascuno di questi circuiti *monitor del bus* per il bus dati bidirezionale del microprocessore 8080.

Si possono eseguire operazioni di latch su dati in ingresso, dati in uscita, dati letti in memoria, dati scritti in memoria e vettori di interrupt, facendo sì che sia applicata l'opportuna linea di controllo all'ingresso di abilitazione del latch del circuito di monitor del bus.

ESPERIMENTO 1-2
CIRCUITO SINGLE-STEP

Scopo

Lo scopo di questo esperimento è di costruire un circuito single-step per un microcomputer basato sull'8080A.

Configurazione di pin del chip del circuito integrato (Fig. 1-9)

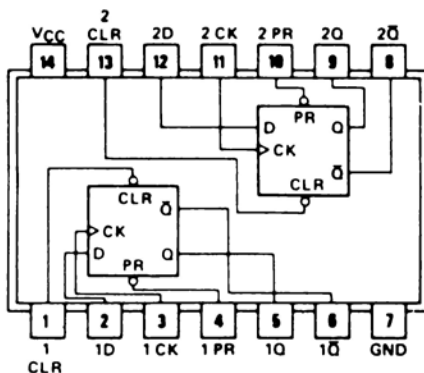



Fig. 1-9. Configurazione del pin del 7474.

Schema del circuito (Fig. 1-10)

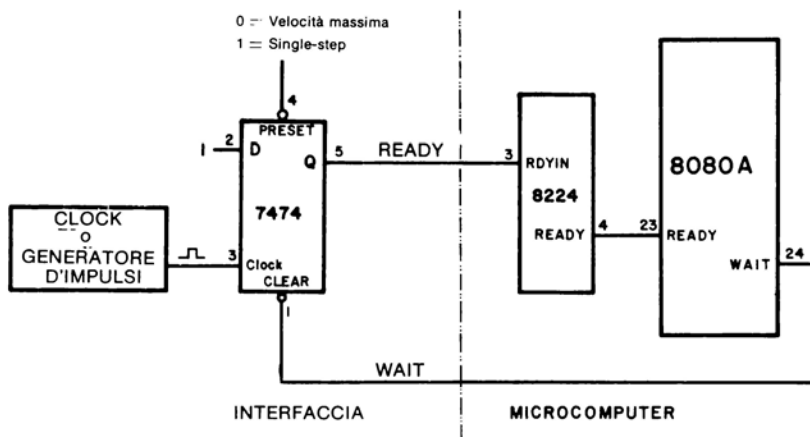



Fig. 1-10. Circuito per single-step.

Programma (Fig. 1-11)

```
              /
              /PROGRAMMA DI TEST DEL SINGLE STEP
              /
              *003 000
003 000 000   START, NOP     /NESSUNA OPERAZIONE
003 001 074          INRA    /INCREMENTA DI UNO L'ACCUMULATORE
003 002 323          OUT     /CONTENUTO DELL'ACCUMULATORE IN USCITA
003 003 000          000     /CODICE DISPOSITIVO DELLA PORTA D'USCITA
003 004 303          JMP     /SALTO INCONDIZIONATO
003 005 000          START   /LO INDIRIZZO DI MEMORIA DEL SALTO
003 006 003          0       /HI INDIRIZZO DI MEMORIA DEL SALTO
```

Fig. 1-11. Programma per il circuito di single-step.

Passo 1

Collegare il circuito mostrato in Fig. 1-10. Se si sta usando un microcomputer Mini-Micro Designer (MMD-1), si possono trovare le linee READY e WAIT sulla piastra. Altrimenti, si dovranno sistemare queste linee di controllo sul proprio microcomputer. Collegare il circuito monitor del bus al bus dati e la linea di abilitazione latch a massa. Il latch sarà così permanentemente abilitato e il bus monitor mostrerà continuamente i dati che compaiono sul bus dati.

Passo 2

Inserire il programma riportato in Fig. 1-11 nella memoria di lettura/scrittura.

Passo 3

Iniziare l'esecuzione del programma alla *massima velocità* del microcomputer. Che cosa vedete sul monitor bus? Perchè?

Abbiamo visto che tutti i LED del monitor bus, erano luminosi (8 su un display a sette-segmenti). La ragione di ciò è che il bus monitor, sta mostrando le istruzioni di macchina e i byte di dati, non appena questi appaiono sul bus dati durante l'esecuzione del programma. Alla massima velocità del microcomputer, questi byte cambiano così rapidamente che i LED del monitor bus appaiono permanentemente accesi.

Passo 4

Per controllare che il circuito single-step funzioni correttamente, passare in *single-step* e iniziare a scorrere il programma *un ciclo di macchina* alla volta, azionando il pulsante di single-step. Scrivere, nello spazio qui sotto, i byte osservati fino a che non emerge una sequenza chiara.

La sequenza di byte osservata era:

000, 074, 323, 000, ***, 303, 000, 003

dove il byte *** rappresenta il byte dati che si sta portando fuori dall'accumulatore. Il suo valore cambia non appena il programma viene eseguito.

ESPERIMENTO 1-3
UNA PORTA D'INGRESSO IN MAPPA DI MEMORIA

Scopo

Lo scopo di questo esperimento è di collegare ed esaminare il funzionamento del chip circuito-integrato 8212 (SN74S412), come una porta in mappa di memoria.

Configurazione dei pin e tabella della verità dell'8212 porta Ingresso/Uscita a otto-bit.

NOMI DEI PIN

DI1 DI8	INGRESSO DATI
DO1 DO8	USCITA DATI
DS1 DS2	SELEZIONE DISPOSIT.
MD	MODO
STB	STROBE
INT	INTERRUPT (ATTIVO BASSO)
CLR	CLEAR (ATTIVO BASSO)

STB	MD	(DS1·DS2)	USCITA DATI UGUALE A
0	0	0	3 STATE
1	0	0	3 STATE
0	1	0	LATCH DATI
1	1	0	LATCH DATI
0	0	1	LATCH DATI
1	0	1	INGRESSI DATI
0	1	1	INGRESSI DATI
1	1	1	INGRESSI DATI

CLR - RESET LATCH DEI DATI
SET DEL FLIP-FLOP SR
(NESSUN EFFETTO SUL BUFFER D'USCITA)

CLR	(DS1·DS2)	STB	*SR	INT
0	0	0	1	1
0	1	0	1	0
1	1	↓	0	0
1	1	0	1	0
1	0	0	1	1
1	1	↓	1	0

* FLIP-FLOP SR INTERNO

Concessione dell'Intel Corp.

Fig. 1-12. Configurazione dei pin della porta I/O e tabella della verità.

Schema del circuito (Fig. 1-13)

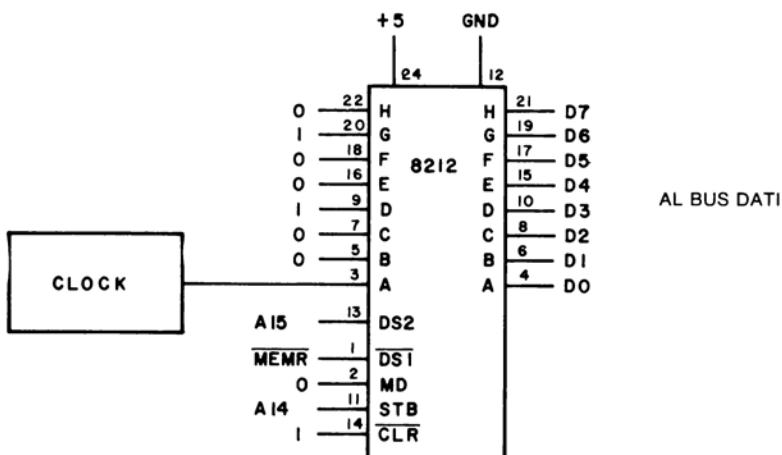



Fig. 1-13. IC 8212 come porta d'ingresso in mappa in memoria.

Programma (Fig. 1-14)

```
                       /
                       /PROGRAMMA DI TEST PER I/O
                       /IN MAPPA DI MEMORIA
                       /
                       DB BUSMON  000
                       DW MMCODE 300 000
                       *030 000
0030 000 041                LXIH      /CARICA IN H,L IL CODICE DI SELEZ
                                      /INDIRIZZO
030 001 000                 MMCODE    /LO BYTE INDIRIZZO
030 002 300                 0         /HI BYTE INDIRIZZO
030 003 176          LOOP,  MOVAM     /CARICA IN A IL CONTENUTO DELLA
                                      /LOCAZIONE "M"
030 004 323                 OUT       /A IN USCITA ALLA PORTA 0
030 005 000                 BUSMON    /CODICE DISPOSITIVO DELLA PORTA 0
030 006 303                 JMP       /SALTA AL LOOP A
030 007 003                 LOOP      /L0 BYTE INDIRIZZO
030 010 030                 0         /HI BYTE INDIRIZZO
```

Fig. 1-14. Programma per una porta d'ingresso in mappa di memoria.

Passo 1

Caricare in memoria il programma. Inserire 300_8 all'indirizzo di memoria 003_8002_8.

Passo 2

Collegare il circuito bus monitor, dato nell'Esperimento 1-1, a questa unità. Lo schema a blocchi per un tal circuito, sarà dato in esperimenti successivi, come indicato in Fig. 1-15.

Fig. 1-15. Schema del circuito monitor del bus.

Passo 3

Collegare il circuito dato in Fig. 1-13. La Fig. 1-16 è la nota applicativa per l'impiego del circuito integrato 8212, come un gate buffer d'ingresso three-state.

Vengono di seguito riportati i pin di controllo dell'8212, le linee di controllo collegate a tali pin ed i livelli logici necessari per l'ingresso dati.

Ingresso 8212	$\overline{DS1}$	DS2	MD	STB	$\overline{CLR}$
Linea di Controllo	$\overline{MEMR}$	A15	0	A 1 4	0
Livello Logico per Ingresso Dati	0	1	0	1	1

La nota applicativa (Fig. 1-16) e la tabella della verità dell'8212 (Fig. 2-12), confermano che, ai pin di controllo dell'8212, devono essere applicati i cinque livelli logici sopra riportati per l'ingresso dati. Nello

Gated Buffer (3 - STATE)

Il più semplice impiego dell'8212 è quello come gate buffer. Con il segnale di modo basso e l'ingresso di strobe alto, il latch dei dati si comporta come un gate lineare. I buffer d'uscita vengono poi abilitati dalla logica di selezione dispositivo DS1 e DS2.
Quando la logica di selezione dispositivo è falsa, le uscite sono 3-state.
Quando la logica di selezione dispositivo è vera, i dati inseriti dal sistema vengono trsferiti direttamente in uscita. Il carico per l'ingresso dei dati è di 250 micro ampere. L'uscita dei dati può assorbire 15 milli ampere. Il valore minimo dell'ampiezza di uscita è 3,65 Volt.

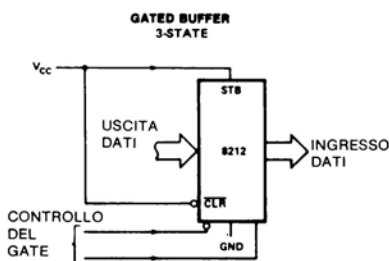



Fig. 1-16. Nota applicativa per l'8212.

spazio qui sotto, scrivere l'indirizzo di memoria richiesto per abilitare l'8212 all'ingresso dati. Usare una X per i bit "non significativi" ("don't care" bits).

Se sostituite i bit X con il valore logico 0, l'indirizzo di memoria ottenuto è coerente con il nostro codice selezione indirizzo, inserito al L0 byte dell'indirizzo 001_8 e 002_8 del programma introdotto al Passo 1?

L'indirizzo di memoria richiesto per abilitare l'8212 allo ingresso dati è

$$11\ XXX\ XXX_2\ XX\ XXX\ XXX_2$$

Sostituendo i bit non significativi con il valore logico 0, si ha 300_8000_8, che è l'indirizzo dato al L0 byte dell'indirizzo 001_8 e 002_8 del programma.

Passo 4

Mettere il clock dell'oscillatore, a circa 10 Hz ed eseguire il programma alla piena velocità del microcomputer. Quali byte ottali osservate sul monitor bus?

Noi vediamo i byte 110_8 e 111_8, con il digit ottale meno significativo che cambia alla stessa frequenza del clock dell'oscillatore.

Passo 5

Sostituire a turno, nel programma, il HI byte dell'indirizzo, alla locazione di memoria 003_8002_8, con i seguenti byte: 010_8 e 365_8. Descrivere l'osservazione del comportamento del monitor bus durante ogni esecuzione di programma.

Noi abbiamo osservato, un normale trasferimento dati dall'ingresso al monitor bus, con il HI byte dell'indirizzo 365_8 quando si abilita la porta d'ingresso (A15 e A14 al valore logico 1). Si vede 377_8 sul monitor bus, quando il programma viene eseguito con 010_8 alla locazione di memoria 003_8 002_8. In questo caso il bit d'indirizzo A15 è zero e quindi le linee d'ingresso dell'8212 sono nel modo three-state e non viene inserito alcun dato.

Passo 6

Il bus monitor collegato come in questo esperimento, viene impiegato come porta d'uscita. Il trasferimento dati a questa porta, avviene con la tecnica di I/O in mappa di memoria oppure in accumulatore? Qual'è

l'inconveniente di questa porta che si vede subito?

Il bus monitor è, certamente, una porta d'uscita dell'accumulatore, dal momento che si usa l'impulso OUT per abilitare i latch durante l'uscita dei dati dall'accumulatore. Il problema della porta, che si può subito rilevare, è che non possiede un codice dispositivo unico. Si può avere conferma di ciò cambiando, con un altro valore, "BUSMON" alla L0 locazione di memoria 005_8 nel programma. (NOTA: Se si usa un microcomputer MMD-1, non sfruttare i codici dispositivo 000_8, 001_8, 002_8, poichè questi sono già impiegati per le tre porte d'uscita dei LED).

Conservare questo circuito per l'esperimento successivo.

ESPERIMENTO 1-4
ESECUZIONE DELL'OPERAZIONE AND

Scopo

Lo scopo di questo esperimento è di mostrare l'esecuzione di un'operazione AND fra una porta d'ingresso in mappa di memoria e l'accumulatore.

Schema del Circuito

Usare il circuito collegato nell'Esperimento 1-3.

Programma (Fig. 1-17)

```
                    DB BUSMON 000
                    *003 000
003 000 041               LXIH      /CARICA LA COPPIA DI REGISTRI H CON:
003 001 003               003       /L0 BYTE INDIRIZZO DELLA PORTA
                                    /D'INGRESSO
003 002 300               300       /HI BYTE INDIRIZZO DELLA PORTA
                                    /D'INGRESSO
003 003 076         TEST, MVIA      /METTERE I SEGUENTI BYTE IN
                                    /ACCUMULATORE
003 004 001               001       /BYTE MASCHERA
003 005 246               ANAM      /AND FRA IL CONTENUTO DELLA PORTA
                                    /D'INGRESSO E L'ACCUMULATORE
003 006 312               JZ `      /IL RISULTATO È ZERO? SI - SALTA
                                    /INDIETRO
003 007 003               TEST      /A TEST
003 010 003               0
003 011 076               MVIA      /NO, FLAG BIT = 1 QUINDI CARICA IN A
003 012 377               377       /TUTTI UNO
003 013 323               OUT       /USCITA DEL BYTE
003 014 000               BUSMON    /ALLA PORTA 0
003 015 166               HLT       /ARRESTO DEL MICROCOMPUTER
```

Fig. 1-17. Programma per l'operazione AND.

Passo 1

Montare sulla piastra, se non lo è già, il circuito porta d'ingresso in mappa di memoria, mostrato nell'esperimento 1-3 (Fig. 1-13).

Passo 2

Caricare ed eseguire il programma di Fig. 1-17 con l'oscillatore del clock, posto a circa 2Hz. Che cosa succede alla porta d'uscita BUSMON?

Noi, sul monitor del bus, vediamo il codice ottale 337.

Passo 3

Porre i valori 111 e 276 nei L0 byte degli indirizzi di memoria 004 e 005. Il secondo dei valori inseriti, è il codice operativo dell'istruzione CMP M. Eseguire ora il programma alla massima velocità. Annotare e spiegare, nello spazio qui sotto, ciò che si osserva sul monitor del bus.

Noi sul monitor, vediamo 377. L'istruzione CMP M, confronta il contenuto dell'accumulatore (111 in questo caso) con il contenuto della locazione di memoria M (in questo caso la porta d'ingresso in mappa di memoria). Se i due byte sono uguali, viene posto ad 1 il flag zero ed il programma salta a TEST. Se i due byte non sono uguali, viene messo 377 sul monitor del bus.

Domande

1. Qual'è il vantaggio dell'I/O in mappa di memoria rispetto all'I/O in accumulatore, in questa operazione di mascheratura?

2. Scrivere un programma impiegando l'I/O in accumulatore, che ripeta quanto fatto in questo esperimento. Confrontare il numero di byte richiesto in ognuno dei due casi.

ESPERIMENTO 1-5
CONFRONTO FRA I/O IN ACCUMULATORE ED I/O IN MAPPA DI MEMORIA

Scopo

Lo scopo di questo esperimento è di mostrare e confrontare il comportamento delle due diverse tecniche di I/O, cioè, *I/O in accumulatore* e *I/O in mappa di memoria*, per l'ingresso di dati in un microcomputer 8080.

Configurazione dei pin dei circuiti integrati (Fig. 1-18)

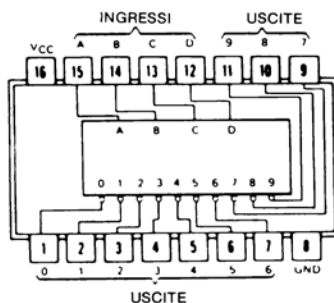
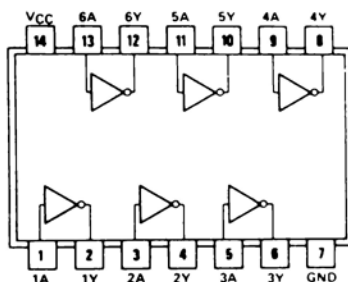
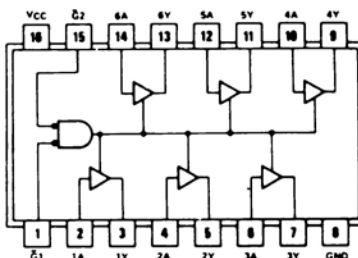



Fig. 1-18. Configurazione dei pin dell'IC.

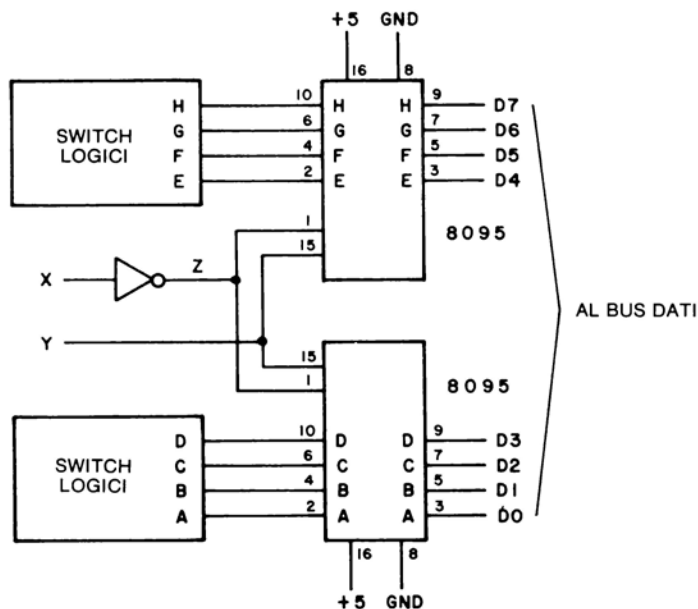
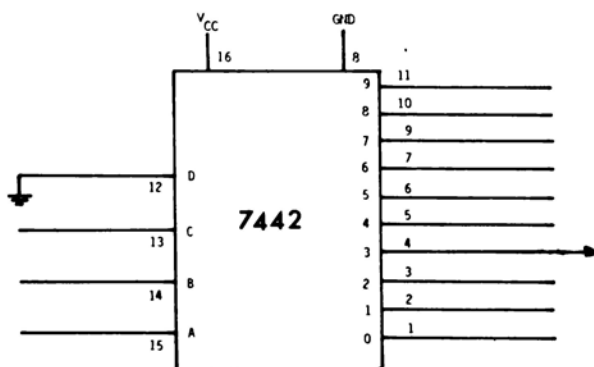
Schemi dei circuiti (Fig. 1-19)

(A) Circuito Buffer d'ingresso

(B) Circuito di decodifica dell'impulso di selezione dispositivo

Fig. 1-19. Schemi dei circuiti.

Programma N. 1. I/O in accumulatore (Fig. 1-20)

```
                 DB BUFF 200
                 *003 000
003 000 333     START, IN        /INSERIRE IN A I DATI DAL
003 001 200            BUFF      /CIRCUITO BUFFER
003 002 323            OUT       /BYTE DATI IN USCITA ALLA
003 003 000            000       /PORTA 0
003 004 303            JMP       /SALTO INCONDIZIONATO A
003 005 000            START     /L0 BYTE INDIRIZZO DI MEMORIA
003 006 003            0         /HI BYTE INDIRIZZO DI MEMORIA
```

Fig. 1-20. Programma N° 1.

Programma N. 2. I/O in mappa di memoria (Fig. 1-21)

```
                 DW MMCODE 300 000
                 *003 200
003 020 041            LXIH      /CARICA LA COPPIA DI REGISTRI H CON:
003 021 000            MMCODE    /CODICE DEL DISPOSITIVO I/O
003 022 300            0         /IN MAPPA DI MEMORIA
003 023 176     LOOP,  MOVAM     /METTERE IL CONTENUTO DELLA
                                 /LOCAZIONE DI MEMORIA M
                                 /DATO DALLA COPPIA DI REGISTRI H IN A
003 024 323            OUT       /CONTENUTO DI A IN USCITA ALLA
003 025 000            000       /PORTA 0
003 026 303            JMP       /SALTA A
003 027 023            LOOP      /L0 BYTE INDIRIZZO
003 030 003            0         /HI BYTE INDIRIZZO
```

Fig. 1-21. Programma N° 2.

I/O in accumulatore

Passo 1

Montare il circuito di buffer d'ingresso indicato in Fig. 1-19A. Collegare la linea di controllo $\overline{IN}$ al punto Y e porre il punto X al valore 1 logico.

Passo 2

Caricare in memoria il Programma N. 1. Porre il byte BUFF, codice del circuito buffer, al valore 220_8 nella locazione di memoria 003_8 001_8.

Passo 3

Eseguire il programma alla massima velocità del microcomputer. Variare gli switch logici all'ingresso del circuito buffer e spiegare nello spazio qui sotto, che cosa si osserva alla porta 0.

Noi abbiamo visto che il byte della porta 0 visualizzato, era quello stesso inserito sugli switch logici.

Passo 4

Cambiare il byte BUFF, codice del circuito di Buffer, inserendo 350_8. Far eseguire il nuovo programma. Che cosa si può osservare quando vengono variati gli switch logici?

Noi abbiamo visto che il byte visualizzato sul monitor del bus, era ancora uguale a quello posto sugli switch logici. Spiegare, nello spazio qui sotto, perchè il codice del dispositivo non ha alcuna influenza.

La ragione è che, per il circuito collegato nel Passo 1, l'impulso di abilitazione buffer è $\overline{IN}$, il quale é generato ogni volta che viene eseguita un'istruzione IN <B2>. Per il codice dispositivo, per abilitare i buffer three-state occorre che:

- uno dei bit (A0-A7) del byte codice dispositivo <B2>, il quale compare sulle linee indirizzi durante un'istruzione $\overline{IN}$, sia unito con $\overline{IN}$, per produrre un *impulso selezione dispositivo,* oppure

- alcuni o tutti i bit A0-A7, siano decodificati per generare un impulso, il quale, ancora unito con $\overline{IN}$, produca un impulso di selezione dispositivo.

In tal caso si può utilizzare un gate NOR, disponibile sul chip 8095, ed è proprio quello che faremo.

Passo 5

Togliere il valore logico 1 dal punto X e collegare tale punto alla linea indirizzo A7 del buffer d'ingresso. Con riferimento allo schema di configurazione dei pin dell'8095 ed al circuito del buffer d'ingresso, si può vedere che si hanno ora $\overline{A7}$ e $\overline{IN}$, messi in NOR. La conseguenza di ciò è un'uscita logica ai buffer three-state, che è normalmente bassa (alta impedenza) e che va alta (abilitato) quando entrambi $\overline{A7}$ e $\overline{IN}$ sono bassi.

Questa ultima condizione, si verifica solamente durante il terzo ciclo di macchina di un'istruzione IN <B2>. Da qui si ha che il codice dispositivo per il circuito buffer d'ingresso sarà

A7	A6	A5	A4	A3	A2	A1	A0	=	1	X	X	X	X	X	X	X

dove X = "non significativo"

Passo 6

Ripetere il Passo 3 con BUFF, alla locazione di memoria 003_8 001_8, posto rispettivamente ai valori 100_8, 220_8 e 350_8 e spiegare perchè i dati, in seguito a ciò, vengono trasferiti dagli switch logici alla porta 0 d'uscita negli ultimi due casi, ma non nel primo.

Se trovate difficoltà nel rispondere a tale domanda, rivedete il commento di Passo 5 e osservate che il codice dispositivo nel Passo 6, ha il bit d'indirizzo A7 al valore logico 1.

Passo 7

Chiaramente, si ha una considerevole ridondanza del codice dispositivo dall'impiego di un solo bit di codice indirizzo, per generare un impulso di selezione dispositivo. Una parte o tutta, tale ridondanza, può essere eliminata, decodificando alcuni o tutti, i bit codice di dispositivo A0-A7 (o A8-A15). Per chiarire ciò, togliere potenza dalla piastra e montare il circuito di decodifica SN7442, riportato in Fig. 1-19B, preferibilmente sulla stessa piastra, come circuito buffer d'ingresso. Collegare le linee indirizzi A4, A3 e A2, rispettivamente agli ingressi C, B e A del SN7442. Collegare la uscita "3" del SN7442 (pin 4), direttamente al punto Z del circuito buffer d'ingresso e sconnettere A7 dal punto X. Accertarsi di aver sconnesso da Z anche l'inverter. Quali codici dispositivo abiliteranno ora il circuito buffer d'ingresso?

Qualsiasi codice che ha A4 al valore logico 0 e A3, A2 entrambi al valore logico 1, abiliterà il buffer d'ingresso.

Passo 8

Porre BUFF al valore 00001111 = 017, ripetere il Passo 4 e constatare che i dati vengono successivamente trasferiti dagli switch logici alla porta 0, una volta alimentata la piastra.

I/O in mappa di memoria

Passo 9

Per predisporre il circuito buffer d'ingresso per l'I/O in mappa di memoria, togliere l'alimentazione alla piastra, sconnettere $\overline{\text{IN}}$ al punto Y e collegare $\overline{\text{MEMR}}$ al suo posto. Sconnettere anche l'uscita del decodificatore dal punto Z e collegare la linea indirizzo A15 al punto X. Ricordarsi di ricollegare al punto Z anche l'inverter. Il bit indirizzo A15, indicherà ora se il microcomputer sta accedendo alla memoria (A15 al valore logico 0) o al dispositivo I/O (A15 al valore logico 1).

Passo 10

Caricare il programma N. 2 in memoria.

Passo 11

Alimentare la piastra ed iniziare l'esecuzione del programma. Dall'esecuzione del programma, si ha conferma che i dati inseriti dalla porta d'ingresso in mappa di memoria, vengono visualizzati ancora sulla porta 0.

Passo 12

Come per l'I/O in accumulatore, anche in questo caso, esiste una considerevole duplicazione del codice dispositivo, quando vengono impiegati il bit indirizzo A15 e $\overline{\text{MEMR}}$, poichè ogni indirizzo superiore a 200_8000_8 abiliterà il buffer d'ingresso. Constatare che si verifica ciò sostituendo diversi codici di selezione indirizzo nelle locazioni di memoria 003_8021_8 e 003_8002_8.

Passo 13

Come nel caso dell'I/O in accumulatore, anche qui si può ridurre la ridondanza del codice di dispositivo, impiegando un circuito di decodifica. Tracciare un circuito nello spazio sotto a quello riservato agli indirizzi di memoria sotto 48K e agli indirizzi di I/O sopra 48K. Si può voler controllare tale circuito.

Il circuito richiesto richiede A15 ed A14 collegati agli ingressi di un gate NAND a due ingressi, il quale sostituisce l'inverter al punto Z di Fig. 1-19A.

Passo 14

Che cosa bisogna fare se si vuole inserire un byte ad 8 bit dagli switch logici nel registro C, anzichè nell'accumulatore? Quali modifiche sono necessarie per fare i programmi per l'I/O in accumulatore e l'I/O in mappa di memoria, che permettono tale operazione? Scrivere i nuovi programmi nello spazio qui sotto.

Per il programma di I/O in accumulatore, bisogna aggiungere un'istruzione MOV C,A alla locazione 003 002. Per il programma di I/O in mappa di memoria, bisogna sostituire l'istruzione MOV A, M con la MOV C, M.

Passo 15

Sulla base di questi ultimi programmi, individuare un vantaggio ed uno svantaggio dell'I/O in mappa di memoria.

Il vantaggio dell'I/O in mappa di memoria rispetto all'I/O in accumulatore, illustrato da tali programmi, è che, con lo I/O in mappa di memoria, i dati possono essere posti direttamente nel registro desiderato, mentre con l'I/O in accumulatore è necessaria un'istruzione MOV extra. Tale risparmio di un'istruzione è utile dove si devono trasferire ed elaborare grossi blocchi di dati. L'inconveniente dell'I/O in mappa di memoria, che è qui illustrato, è la maggior lunghezza del programma e la necessità di vincolare, per l'I/O, la coppia di registri H, L.

CAPITOLO 2

UN'ANALISI DELL'8255

2-1. ELETTRONICA

In questo paragrafo esamineremo i blocchi elettronici funzionali, di base, interni al PPI, per vedere come tali blocchi interagiscono con il mondo esterno, attraverso le loro linee di controllo e dati. Il nostro scopo è quello di esaminare il rendimento del PPI come interfaccia I/O e vedere anche i collegamenti hardware necessari per un soddisfacente funzionamento del chip.

La Fig. 2-1, mostra un diagramma a blocchi del PPI. Si osservi che il PPI può essere diviso in tre unità principali, cioè, i circuiti d'interfaccia all'unità di elaborazione centrale 8080 (CPU), un'unità d'interfaccia periferica, un'unità di controllo della logica interna.

(A) Pin interfaccia periferica

Poichè oltre il 50 per cento dei pin dell'8255 sono riservati ad operazioni di ingresso o uscita dati, parleremo innanzi tutto di questi pin. I dati vengono trasferiti a e da dispositivi esterni di I/O, per mezzo di tre porte ad 8 bit, conosciute come porta A (PA0-PA7), porta B (PB0-PB7) e porta C (PC0-PC7).

Le funzioni e le caratteristiche di queste tre porte, sono determinate dal modo di funzionamento del PPI, il quale è selezionato sotto controllo del programma. Esistono tre modi in cui le linee I/O dell'8255 possono essere programmate:

- *Ingresso base e uscità base (Modo 0)*. In questo modo le 24 linee I/O vengono suddivise in due gruppi di otto linee ciascuno (porte A e B) e due gruppi di quattro linee che insieme vengono chiamati porta C. Ogni porta o gruppo, può poi essere programmato individualmente per operazioni di *base* ingresso o uscita. In questo modo di funzionamcnto, che è il più semplice dei tre, ogni designata porta d'uscita è sottoposta a latch. Le porte d'ingresso non sono sottoposte a latch, dal momento che funzionano come buffer d'ingresso three-state.

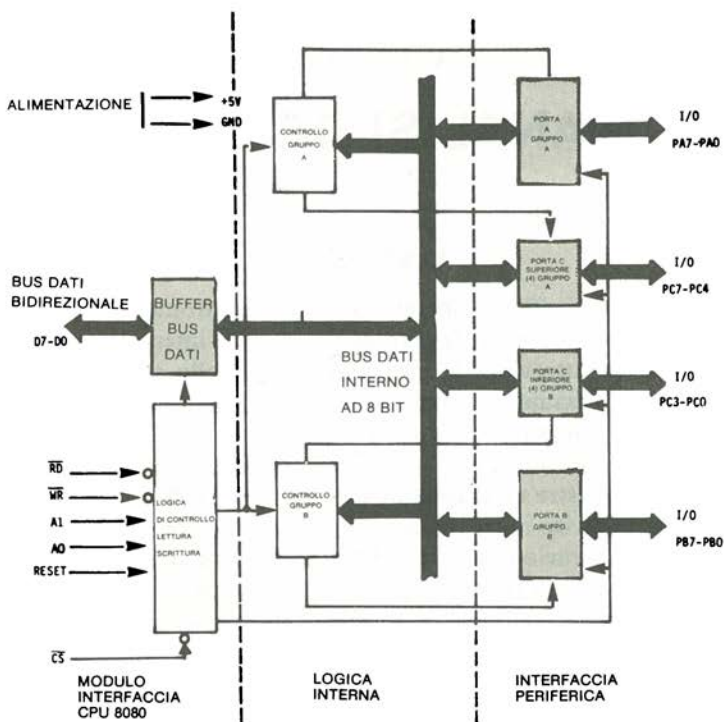



Fig. 2-1. Schema a blocchi dell'8255 PPI.

- *Ingresso e uscita con strobe (Modo 1).* In questo modo il PPI usa le due porte ad 8 bit, A e B, come porte d'ingresso od uscita unidirezionali. Ogni volta trasferisce i dati congiuntamente con un segnale di strobe o "handshaking". Le porte A e B, sfruttano gli 8 bit della porta C, per generare o accettare questi segnali di "handshaking". Sulle porte A e B, i dati sono sottoposti a latch sia in ingresso che in uscita.
- *I/O Bidirezionale con strobe (Modo 2).* Nel funzionamento in modo 2, è disponibile una sola porta I/O bidirezionale (porta A). Come nel modo 1, vengono impiegati segnali di strobe o di "handshake", per mantenere un ordinato flusso di dati dalla e, verso, la CPU. Cinque bit della porta C, vengono impiegati per tale scopo.

Tabella 2-1. Collocazione delle linee I/O nel funzionamento in Modo 0,1 e 2			
Modo	**Porta A**	**Porta B**	**Porta C**
0	INGRESSO/USCITA BASE Uscite con latch Ingressi senza latch	INGRESSO/USCITA BASE Uscite con latch Ingressi senza latch	INGRESSO/USCITA BASE Uscite con latch Ingressi senza latch
1	I/O CON STROBE Ingressi ed uscite con latch	I/O CON STROBE Ingressi ed uscite con latch	CONTROLLO/BIT DI STATO PER LE PORTE A E B
2	I/O BIDIREZIONALE CON STROBE Ingressi ed uscite con latch	—	CONTROLLO/BIT DI STATO PER LA PORTA A

La Tabella 2-1, presenta un sommario dei principali modi in cui vengono collocate le 24 linee I/O, per i tre maggiori modi di funzionamento. Quando si programma il modo di funzionamento del PPI, non si è costretti ad assegnare una particolare regola di funzionamento a *tutte* le linee I/O del PPI. Si fissa la *parola di controllo modo*, che definisce il modo di funzionamento di ogni porta PPI, cosicchè, ad ogni porta può essere assegnato un diverso modo di funzionamento. Se ad esempio, il funzionamento della porta A è programmato in modo 2, le rimanenti otto linee della porta B e le tre linee della porta C, possono essere configurate per funzionare sia in modo 0 che in modo 1.

Un'ulteriore caratteristica della porta C, è che ogni bit può essere individualmente posto a 0 o a 1. Ciò è importante perchè permette di generare segnali di strobe e di gate da software, impiegando una *parola di controllo set/reset-bit*. Tale fatto permette di eliminare la necessità di logica esterna aggiuntiva, sebbene si richiedano passi di software in più per generare le condizioni di set/reset.

(B) Pin interfaccia della CPU 8080A

Con riferimento alla Fig. 2-1, ci sono otto linee dati, sei linee di controllo e due linee di alimentazione che costituiscono i rimanenti pin del chip PPI. Tali pin sono relativi all'interfaccia PPI e CPU 8080 (Fig. 2-1). Esaminiamo il compito di ognuno di questi pin.

Durante l'esecuzione dei programmi IN, OUT e MOV ingresso/uscita, la CPU 8080A, comunica con il PPI per mezzo del bus dati del sistema, il quale è collegato ai pin da D0 a D7. Tutti i byte dati che passano fra l'8080 e l'8255, vengono trasmessi o ricevuti per mezzo di un buffer bidirezionale a 8 bit. Ora, in ogni computer ci possono essere molti

dispositivi, considerando memorie di lettura/scrittura (RAM), ROM, EPROM, PPI e USART, collegati al bus dati. Poichè solamente un dispositivo alla volta può essere "attivo", se si vuole evitare il sovraccarico del bus, il PPI, così come gli altri dispositivi, ha le linee del bus dati three-state. Tali linee vengono abilitate, e il chip è effettivamente collegato al bus, per mezzo di un valore logico zero sulla linea di *selezione chip* del PPI ($\overline{CS}$). Un uno logico sul pin di selezione chip, costringe le linee del bus dati del PPI in uno stato di alta impedenza.
Una volta che il PPI è stato abilitato con uno zero logico su $\overline{CS}$, si deve poi dire se esso deve leggere dati ($\overline{RD}$) dalle porte I/O e metterli sul bus dati per l'8080, o deve invece, scrivere i dati ($\overline{WR}$) presenti sul bus (D0-D7) alle porte I/O. Chiaramente lo zero logico contemporaneo su $\overline{RD}$ e $\overline{WR}$, rappresenta una condizione non consentita!

Consideriamo un attimo i tipi di byte che possono essere inviati sul bus dati al PPI. Questi sono costituiti da:

- *Byte dati* per la porta A, porta B, o porta C.
- *Byte di controllo*, che sono inviati ad un registro di controllo all'interno del Blocco di Controllo Logico Lettura/Scrittura dell'8255. Esistono due tipi di byte di controllo: la *parola di controllo modo*, che specifica il modo di funzionamento delle porte da A a C, la *parola di controllo set/reset-bit*, la quale è usata per mettere a uno o azzerare ogni singolo bit della porta C.

Quando si invia un byte di controllo al PPI, impiegando I/O in accumulatore o in mappa di memoria, bisogna inviare contemporaneamente anche un *indirizzo di destinazione* o codice, per specificare il tipo di byte e la sua destinazione. I pin d'indirizzo A0 e A1, vengono usati dal PPI per questo scopo. La tabella 2-2, indica il modo in cui i segnali che sono connessi a questi pin, definiscono il tipo e la destinazione di un byte che si sta trasmettendo fra l'8080 e l'8255.

Tabella 2-2. Impiego dei pin indirizzi A0 e A1

A_1	A_0	Operazioni I/O
0	0	BUS DATI ↔ porta A
0	1	BUS DATI ↔ porta B
1	0	BUS DATI ↔ porta C
1	1	BUS DATI → registro di controllo

Chiaramente questi due bit d'indirizzo non forniscono un'indirizzo unico, quando è disponibile un codice dispositivo a 8 bit per I/O in accumulatore, o quando è disponibile un codice di indirizzo a 16 bit per I/O in mappa di memoria. Per il momento è sufficiente dire che un indirizzo non unico non introduce, in genere, difficoltà.

Il pin di reset (RESET), è di solito collegato alla linea di reset del microcomputer. Un 1 logico su questo ingresso, azzera tutti i registri interni, compreso il registro della parola di controllo, selezionando per tutte le porte I/O il loro modo d'ingresso. Quest'ultima caratteristica può essere utile per l'inizializzazione del sistema. Infine, il PPI, richiede una singola alimentazione a +5-V (Vcc e GND) e ciò lo rende facilmente adattabile a, praticamente, tutti i sistemi a microcomputer.

(C) Logica di controllo interno

Come detto in precedenza, i modi di funzionamento delle porte da A a C, così come il funzionamento set/reset-bit della porta C, vengono controllati inviando al PPI, o un byte di controllo *modo*, oppure un byte di controllo *set/reset-bit,* sotto il controllo del software. La destinazione della parola di controllo è il *registro di controllo* (all'interno del blocco di controllo logico lettura/scrittura) il cui codice è A0 = 1, A1 = 1. La logica interna del chip (Fig. 2-1), dirige il trasferimento dei dati e delle informazioni di controllo sul bus dati interno. Il byte di controllo del modo, è trasferito a due controllori di porta, i quali vengono indicati come controllo GROUP A e GROUP B. Il modulo di controllo GROUP A, controlla la definizione del modo (ed il trasferimento dati a e da) della porta A e i quattro bit più significativi della porta C. Allo stesso modo il modulo di controllo GROUP B, controlla la porta B ed i quattro bit meno significativi della porta C.

Il cuore della logica di controllo è, quindi, il registro di controllo ad 8 bit poichè il byte di controllo, che viene scritto in tale registro, definisce in definitiva le caratteristiche di funzionamento del PPI. La Fig. 2-2, mostra il formato dei due tipi di byte di controllo. Il punto più importante da notare a proposito della parola di controllo, è che il bit più significativo (D7), è usato per indicare il tipo di parola di controllo. Se D7 è posto al valore logico 0 (Fig. 2-2B), la parola di controllo sarà usata dal PPI per indicare il bit della porta C che deve essere posizionato. Se D7 è al valore logico 1 (Fig. 2-2A), i rimanenti bit della parola di controllo saranno usati dalla logica interna, per specificare il modo di funzionamento di ogni porta da A a C.

Come esempio di uso della parola di controllo, supponiamo che sia richiesta per il PPI la seguente configurazione:

PORTA A: Uscita in Modo 0	PORTA C: BIT PC7-PC4	Ingresso in Modo 0
PORTA B: Ingresso in Modo 1	PORTA C: BIT PC3-CP0	Uscita in Modo 0

Sfruttando la Fig. 2-2A, i singoli bit della parola di controllo possono essere scelti come segue:

(A) Parola di controllo di definizione del modo (B) Parola di controllo set/reset bit

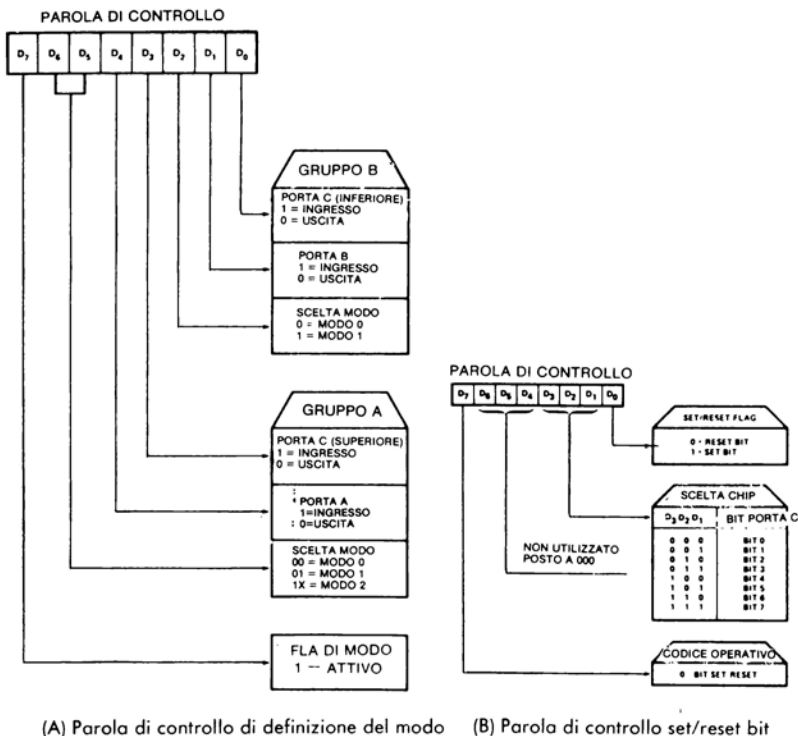



Fig. 2-2. Confronto fra i due tipi di parola di controllo, che definisce le caratteristiche di funzionamento del PPI.

D7	: Codice operativo; modo = 1
D6, D5	: Modo PORTA A; modo 0 = 0,0
D4	: I/O PORTA A; uscita = 0
D3	: I/O PORTA C superiore; ingresso = 0
D2	: Modo PORTA B; modo 1 = 1
D1	: I/O PORTA B; ingresso = 1
D0	: I/O PORTA C inferiore; uscita = 0

La parola di controllo sarà quindi:

1 0 0 0 0 1 1 0 = 206

In maniera simile si può costruire una parola di controllo dalla tabella di Fig. 2-2B e inviarla, usando un'istruzione d'uscita, al registro di controllo per posizionare ciascun bit della porta C da PC7 a PC0. Per azzerare ad esempio il bit PC2, sarà necessaria la parola di controllo

0 0 0 0 0 1 0 1 = 005

Notare che è necessaria una parola di controllo separata, per azzerare PC2, cioè, con ogni parola di controllo set/reset-bit, può essere eseguita una sola operazione di set o reset-bit.

(D) Interfacciamento di un PPI ad un microcomputer

la Fig. 2-3, presenta un approccio per l'interfacciamento di un PPI al bus dati di un microcomputer basato sull'8080, per I/O in accumulatore. Notare che il pin di selezione chip, $\overline{CS}$, è collegato ad A7 attraverso un inverter. L'8255 è quindi abilitato, quando la linea d'indirizzo A7 è al valore logico alto ($\overline{CS}$ basso). Il formato del codice indirizzo di selezione dispositivo, per l'esempio di Fig. 2-3, è illustrato in Fig. 2-4. Gli indirizzi costituiscono il secondo byte delle istruzioni IN ed OUT, usate, in questo esempio, per accedere al PPI. Poichè ci sono molti bit "non significati-

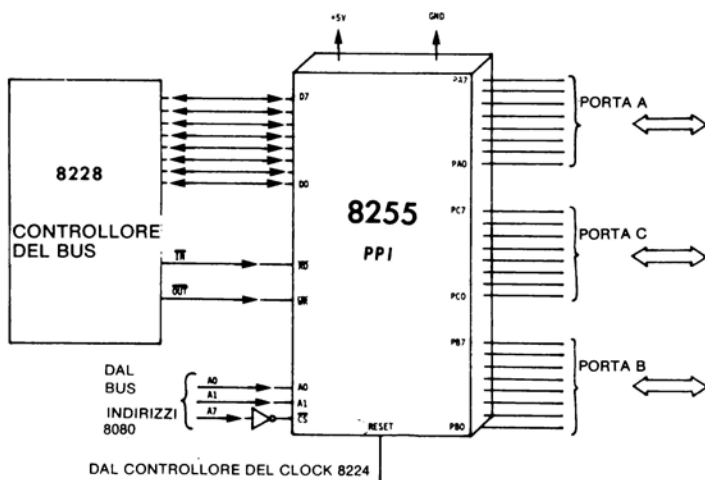



Fig. 2-3. Schema tipico di collegamento dell'8255 PPI per I/O in accumulatore.

vi", esiste una considerevole ambiguità d'indirizzamento e bisogna quindi porre attenzione per evitare di abilitare l'accesso contemporaneo al bus del PPI e di altri dispositivi I/O. Il sovraccarico del bus che si avrebbe, renderebbe non operativo il microcomputer e potrebbe danneggiare l'8080.

La tecnica che impiega un solo bit per abilitare una periferica, è conosciuta come *selezione lineare del dispositivo.* Essa viene usata in piccoli sistemi, come l'SDK-80 della Intel, dove il numero di periferiche è basso ed i rischi e problemi, come spiegato sopra, sono bassi. In sistemi più grandi, come l'Intel SBC-80/10, viene impiegata la decodifica completa della selezione chip, per eliminare l'indirizzamento non univoco del dispositivo. L'autore consiglia di fare quanto sopra anche per piccoli sistemi, dal momento che questi in genere crescono!
Un decodificatore da quattro a 16 linee, 74154, è di solito abbastanza adeguato per decodificare le linee di indirizzo A7, A6, A5 e A4 e fornire un impulso di selezione chip $\overline{CS}$, univoco.

Per l'interfacciamento del PPI di Fig. 2-3 con I/O in mappa di memoria, devono essere fatte, al circuito, le seguenti semplici modifiche:

- Sostituire i segnali di controllo $\overline{IN}$ ed $\overline{OUT}$, rispettivamente con i segnali di controllo di lettura e scrittura dell'I/O in mappa di memoria, $\overline{MEMR}$ e $\overline{MEMW}$.
- Sostituire A7 con A15, per assicurarsi che gli indirizzi del PPI, come porta I/O in mappa di memoria, siano nella seconda metà della memoria.

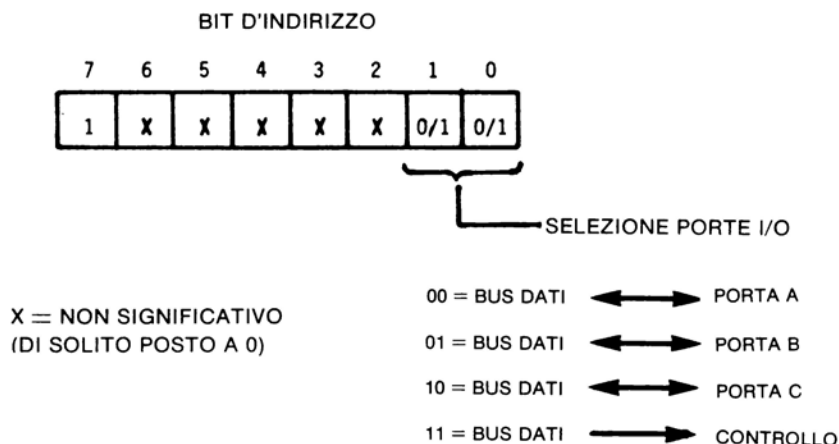



Fig. 2-4. Formato del codice indirizzo di selezione dispositivo per il circuito di Fig. 2-3.

2-2. OPERAZIONI PER L'IMPIEGO DELL'8255

Nell'introduzione abbiamo visto dove conviene il PPI, per un sistema a microcomputer e le ragioni della sua popolarità, cioè, la possibilità di *programmare* il modo di guardare e di comportarsi verso il mondo esterno. Come nella maggior parte dei casi d'impiego di un microcomputer, l'applicazione di questo dispositivo in un sistema a microcomputer, richiede l'uso simultaneo di capacità hardware e software. I tre passi principali richiesti per l'uso del PPI sono:

- Interfacciare il PPI al microcomputer, impiegando I/O sia in accumulatore che in mappa di memoria.
- Programmare il modo di funzionamento dell'8255.
- Ingresso o uscita dati, sotto controllo del software.

Nel precedente paragrafo sull'elettronica del chip, abbiamo visto come si può collegare il PPI sul bus dati dell'8080A, per I/O in accumulatore (Fig. 2-3). È stato anche spiegato il metodo per la costruzione del byte della parola di controllo, per specificare il modo di funzionamento del PPI. [paragrafo 2-1 (C)]. Ciò che rimane, quindi, da discutere per esteso, sono i vincoli software del PPI. I capitoli che seguono tratteranno, pertanto, in dettaglio i requisiti elettronici e di programmazione per i tre modi di funzionamento del PPI.

2-3. SOFTWARE

A parte il modo di funzionamento selezionato per le porte I/O del PPI, il primo passo necessario in ogni programma è una *subroutine di inizializzazione del sistema*. Nel caso più semplice, essa può contenere gli step per inviare al PPI la parola di controllo necessaria a specificare il modo di funzionamento delle tre porte I/O. Supponiamo ad esempio di impiegare il PPI, nella connessione indicata in Fig. 2-3, con i modi di funzionamento delle porte da A a C, specificati come nell'esempio del Paragrafo 2-1 (C). Con riferimento a quel paragrafo, la parola di controllo modo era 206. Per il PPI di Fig. 2-3, l'indirizzo del dispositivo, per il registro di controllo, può essere determinato da Fig. 2-4. Se si pongono i bit "non significativi", da D2 a D6, al valore logico 0, l'indirizzo del dispositivo per il registro di controllo è:

1 0 0 0 0 0 1 1 = 203

Una semplice subroutine d'inizializzazione del PPI, INT, può essere

quindi come quella presentata nel Programma 2-1.

```
                 /
                 /SUBROUTINE D'INIZIALIZZAZIONE DEL PPI: INIT
                 /
                 DB MODE 206
                 DB CNTRL 203
                 *003 200
003 200 365      INIT,   PUSHPSW   /MEMORIZZAZIONE DELLO STATO DEL
                                   /MICROPROCESSORE
003 201 076              MVIA      /CARICARE IN A IL SEGUENTE
003 202 206              MODE      /BYTE DI CONTROLLO DEL MODO
003 203 323              OUT       /USCITA DEL BYTE DI CONTROLLO
                                   /VERSO
003 204 203              CNTRL     /IL REGISTRO DI CONTROLLO
003 205 361              POPPSW    /RIPRISTINARE LO STATO DEL
                                   /MICROPROCESSORE
003 206 311              RET
```

Fig. 2-5. Programma 2-1.

Programma 2-1 (Fig. 2-5)

Nel funzionamento dell'I/O in modo 1 e 2 e a volte in modo 0, alcuni dei bit della porta C devono essere posizionati mediante la parola di controllo set/reset-bit, prima di usare il PPI. In quei casi, anche la parola di controllo set/reset-bit deve essere inviata al registro di controllo del

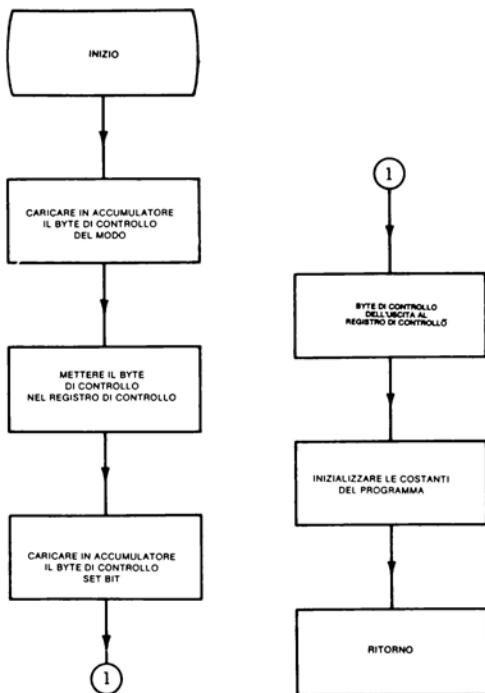



Fig. 2-6. Diagramma di flusso di una subroutine di inizializzazione per un microcomputer contenente un PPI.

PPI, durante l'inizializzazione del sistema. Il diagramma di flusso generalizzato di una routine d'inizializzazione, riportato in Fig. 2, illustra la posizione, all'interno della subroutine, delle istruzioni di set bit necessarie.

Il Programma 2-2 in Fig. 2-7, mostra una subroutine tipica d'inizializzazione del PPI. Notare che, per programmi brevi, i codici d'inizializzazione del PPI, potranno (e più probabilmente dovranno) essere inseriti nel programma principale, per evitare istruzioni PUSH, POP e RETURN.

Programma 2-2 (Fig. 2-7)

Quando si è inizializzato il PPI mediante l'impiego di software, il passo successivo è di trasferire dati attraverso il PPI. Nel caso del PPI di Fig. 2-3, interfacciato per I/O in accumulatore, i dati possono essere inseriti da ogni porta o fatti uscire ad ogni porta, nel modo 0 di base per I/O, impiegando le seguenti istruzioni di I/O in accumulatore:

dove il byte codice dispositivo è l'indirizzo dispositivo delle porte da A a C, come richiesto. Per il circuito di Fig. 2-3, ad esempio, gli indirizzi codice delle porte da A a C, sono dati dalla Fig. 2-4 e sono rispettivamente 200_8, 201_8 e 202_8.

```
                /
                /SUBROUTINE: INITB
                /
                DB MODE 206
                DB CNTRL 203
                DB BITSET 005
                *003 100
003 100 365     INITB,  PUSHPSW  /MEMORIZZARE LO STATO DEL
                                 /MICROPROCESSORE
003 101 076             MVIA     /CARICARE IN ACCUMULATORE LA
003 102 206             MODE     /SEGUENTE PAROLA DI CONTROLLO DEL
                                 /MODO
003 103 323             OUT      /INVIARE LA PAROLA DI CONTROLLO DEL
                                 /MODO AL
```

```
003 104 203      CNTRL    /REGISTRO DI CONTROLLO
003 105 076      MVIA     /CARICARE IN ACCUMULATORE
003 106 005      BITSET   /LA PAROLA DI CONTROLLO SET/RESET
                          /BIT
003 107 323      OUT      /INVIARLA AL
003 110 203      CNTRL    /REGISTRO DI CONTROLLO
003 111 361      POPPSW   /RIPRISTINARE LO STATO DEL
                          /MICROPROCESSORF
003 112 311      RET
```

Fig. 2-7. Programma 2-2.

Nel funzionamento in modo 1 e 2, il trasferimento dati è di solito accompagnato dal risultato del controllo di *impulsi di handshaking*, che sono usati per sincronizzare il trasferimento dati fra un'interfaccia esterna ed il PPI. Il PPI viene innanzi tutto controllato, per vedere se è READY (PRONTO) a ricevere dati dalla CPU da mettere in uscita, o se

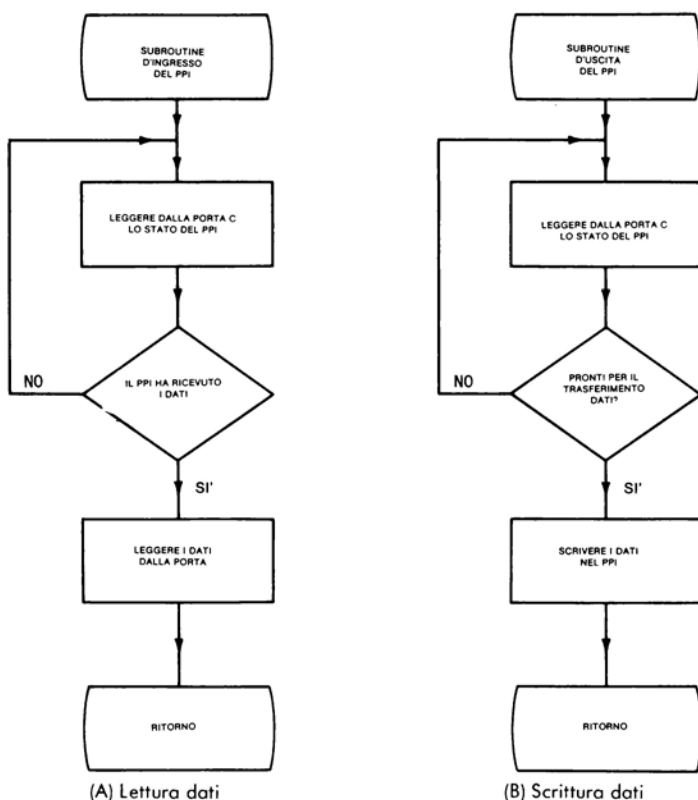



Fig. 2-8. Diagrammi di flusso per il trasferimento dati.

ha ricevuto nuovi dati da inviare alla CPU. Durante il funzionamento in modo 1, le linee della porta C vengono usate come segnali di stato per le porte A e B. Esse devono quindi essere controllate, leggendo la porta C *prima* dell'inserimento o dell'uscita di dati dal PPI. I dettagli di queste operazioni software, saranno discussi nei capitoli succesivi, sul funzionamento nei modi 1 e 2. I diagrammi di flusso di Fig. 2-8, danno un'idea delle tecniche impiegate.

CAPITOLO 3

FUNZIONAMENTO IN MODO 0: I/O SEMPLICE

3-1. INTRODUZIONE

Nei capitoli seguenti esamineremo in dettaglio, i tre modi di funzionamento dell'8255. I modi 0 e 1 del PPI, riflettono, in particolare, due diverse tecniche di *trasferimento dati programmato*, chiamate, trasferimento *incondizionato e condizionato*. Cominciamo definendo e discutendo tali termini. L'apice fra parentesi, fa riferimento alla bibliografia del paragrafo 3-6.

- *Trasferimento dati programmato* [1]: Trasferimento dati fra un sistema a microcomputer e la logica esterna, che è completamente controllata da un programma di microcomputer. La maggior parte delle operazioni ingresso/uscita di un microcomputer, sono trasferimenti dati programmati. Questo tipo di trasferimento dati, non permette alte frequenze di trasferimento, a causa del tempo relativamente basso di esecuzione d'istruzione della maggior parte di microcomputer (almeno 2 μs per l'8080A) e per la necessità di eseguire un certo numero d'istruzioni nel processo d'inserimento o uscita di un byte di dati. Per una migliore frequenza di trasferimento dati, le operazioni ingresso/uscita fra la memoria ed un dispositivo esterno, possono essere ottenute senza interventi del programma, impiegando la tecnica di accesso diretto alla memoria. Tale tecnica è raramente usata in sistema a microcomputer, ad eccezione forse di dispositivi a floppy-disc.
- *Trasferimento dati incondizionato:* [2]: Questo è il tipo più semplice di trasferimento dati e presuppone che il dispositivo esterno sia sempre pronto e disponibile per comunicare con il microcomputer. Esso è del tipo a loop aperto, poichè non c'è feedback dalla periferica per informare il microprocessore, che la periferica è pronta a ricevere i dati, o che i dati sono disponibili sul bus, provenienti dalla

periferica. La tecnica è stata anche descritta come un semplice I/O [1], per indicare il suo basso grado di difficoltà, sia per la programmazione che per l'interfacciamento. Essa è anche stata descritta come *I/O sincrono* [2], presumibilmente per indicare che i dati vengono trasferiti, in sincronismo con il clock del microcomputer e senza far riferimento allo stato di prontezza della periferica. Ad ogni modo l'autore crede che questa ben accettata e compresa logica digitale, non sia applicabile a trasferimento dati parallelo controllato per microcomputer. Si può capire ugualmente bene che, per il fatto che non c'è un clock *comune* al microcomputer e alla logica I/O, la semplice o incondizionata tecnica di I/O è *asincrona*. Per evitare confusione, nell'indicare tale tecnica di I/O, useremo sia "I/O semplice" che "I/O incondizionato".

- *Trasferimento dati condizionato* [3]: In questo approccio, che è spesso chiamato *handshaking I/O* ed è molto comune in sistemi a microcomputer, i dati sono trasferiti a o da, una periferica solamente se essa è pronta per il trasferimento. Tale informazione di stato, sotto forma di flag logico, viene fornita dalla periferica. Questa tecnica può anche essere indicata come *handshaking asincrono di I/O*.

Il modo 0 di funzionamento del PPI, che sarà trattato in questo capitolo, è progettato per il trasferimento *incondizionato* di dati. Quando l'8255 è programmato in modo 0, le tre porte ad 8 bit -porta A, B e C-sono tutte disponibili per questa semplice tecnica di trasferimento dati. Essa è generalmente usata in situazioni in cui una periferica non ha bisogno di indicare al microcomputer, che è pronta per ricevere dati, o che ha dati disponibili da prelevare. Un esempio tipico di una tale situazione, è l'uscita di dati elaborati dal microcomputer, verso vari display alfanumerici. In tali casi i dati possono essere cambiati continuamente senza far riferimento al display. Nel caso di ingresso, ci sono ancora molti casi in cui i dati possono essere inseriti nel microcomputer, senza la necessità di osservare lo stato della periferica in ingresso. Gli 8 bit di un convertitore A/D, che sta convertendo continuamente un segnale d'ingresso analogico, potrà essere inserito nel microcomputer impiegando I/O semplice o incondizionato. Le uscite del SN7493, contatore divisore per 16 funzionante come un contatore di eventi, potranno essere messe in ingresso allo stesso modo.

La discussione sull'I/O in accumulatore ed in mappa di memoria del Capitolo 1, era basata sulla tecnica del trasferimento incondizionato di dati, infatti non si faceva alcun riferimento all'informazione di stato della periferica. Con riferimento alla Fig. 1-4, si può vedere che per l'ingresso, gli elementi essenziali del circuito (Fig. 1-4A), sono gli switch

logici che rappresentano la periferica e due buffer three-state a 4 bit. Chiaramente gli switch logici dovranno in pratica essere sostituiti dal contatore 7493 o da un convertitore A/D, ecc. Il buffer three-state, ad ogni modo, è fondamentale. Per l'uscita anche i latch sono fondamentali, per assicurare che i dati in uscita, durante una operazione d'uscita asincrona, siano trattenuti fino all'aggiornamento con la successiva operazione d'uscita. In accordo con tale necessità dell'I/O semplice, tutte tre le porte del PPI, quando configurate per il funzionamento in modo 0, hanno *latch d'uscita*. Per l'ingresso in modo 0, le otto linee di ogni porta non vengono collegate al bus dati del microcomputer, fino a che la porta non è selezionata per l'ingresso dal microcomputer. Quando si usa il modo 0 per l'ingresso, i dati non subiscono latch dal PPI, ma vengono trasferiti immediatamente per mezzo del PPI al microcomputer. La Fig. 3-1, indica come i circuiti d'ingresso e d'uscita dell'I/O semplice, di Fig. 1-4A e 1-4B, possano essere sostituiti da un solo PPI. Notare che gli otto bit della porta C, sono ancora disponibili per compiti di trasferimento di dati incondizionato.

3-2. REQUISITI

Per impiegare il PPI nella sua configurazione in modo 0, per I/O semplice [3] o incondizionato, sono necessari i seguenti passi (Paragrafo 2-2):

a) Programmare il PPI per il funzionamento in modo 0: questa è l'inizializzazione del sistema.

b) Ingresso e/o uscita dati impiegando I/O semplice.

Questi passi possono quindi richiedere la determinazione di:

- parola di controllo modo per il funzionamento in modo 0 in ingresso o uscita, delle porte da A a C;
- codice dispositivo del registro di controllo;
- indirizzo dispositivo delle porte da A a C.

Come esempio consideriamo il circuito di Fig. 3-1, dove la porta A è stata programmata per il modo 0 in uscita e la porta B per il modo 0 in ingresso. Sebbene la porta C non sia usata, gli assegneremo il modo 0 in uscita. Con riferimento poi alla Fig. 2-2A, i bit da D7 a D0 della parola di controllo modo, devono essere posti come segue:

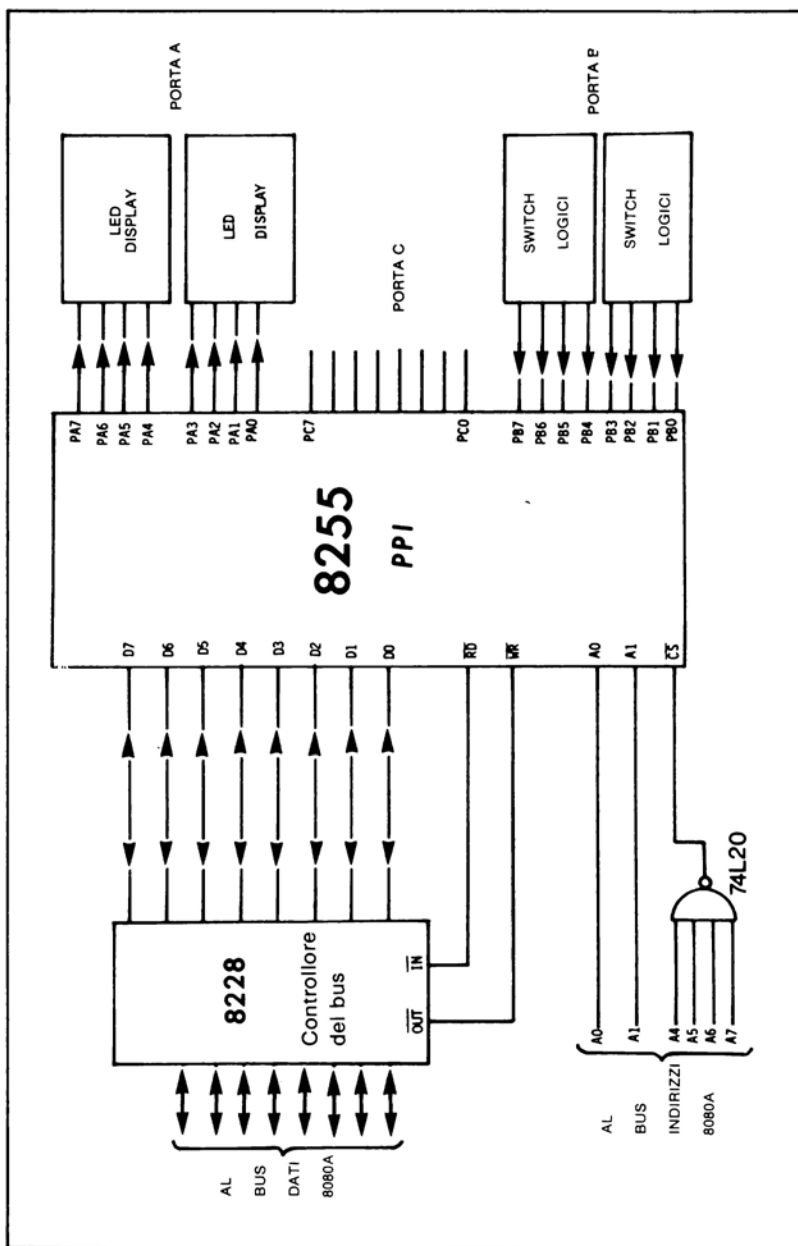



Fig. 3-1. In questo circuito il PPI è collegato come semplice I/O (modo 0) per inserire i dati dalla porta B ed inviarli in uscita sulla porta A.

D7 : codice operativo; modo = 1
D6, D5 : selezione modo gruppo A; modo 0 = 0,0
D4 : I/O porta A; uscita = 0
D3 : **I/O porta C superiore; uscita = 0**
D2 : selezione modo gruppo B; modo 0 = 0
D1 : I/O porta B; ingresso = 1
D0: : **I/O porta C inferiore; uscita = 0**

Oppure

D7,	D6,	D5,	D4,	D3,	D2,	D1,	D0
1	0	0	0	0	0	1	0

202

Segue che la parola di controllo modo richiesta è 202_8. Notare che la porta C è stata divisa in due gruppi di quattro bit, indicati con porta C superiore (PC7-PC4) e porta C inferiore (PC3-PC0); ognuna delle due parti è stata indipendentemente considerata ingresso o uscita. Tale caratteritica della porta C è illustrata nei Passi da 8 a 10 dell'Esperimento 3-1.

Gli indirizzi di dispositivo per le porte da A a C e per il registro di controllo, vengono, in generale, determinati dalle linee d'indirizzo collegate ad A0, A1 e $\overline{CS}$. Nel circuito riportato in Fig. 3-1, le linee indirizzo del microcomputer A0 e A1, sono state collegate ai pin A0 ed A1 del PPI come si fa di solito. Le linee indirizzo da A4 ad A7, sono state collegate insieme a NAND, per fornire una linea di decodifica parziale di selezione chip per l'ingresso $\overline{CS}$. La decodifica parziale usata in questo esempio è

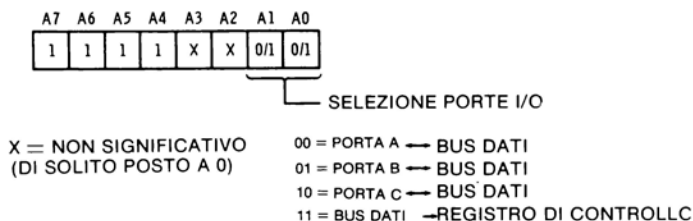



Fig. 3-2. Formato del codice d'indirizzo per la selezione del dispositivo.

superiore alla tecnica di selezione dispositivo lineare impiegata in Fig. 2-3. Ricordando che la linea di selezione chip è *attiva quando posta al valore logico basso* e ricordando la funzione dei pin A0 e A1, come riportato in sommario in Tabella 2-2, si può generare un formato codice d'indirizzo di selezione dispositivo, per il circuito di cui sopra, come indicato nello schema di Fig. 3-2.

Dall'informazione di Fig. 3-2 e ponendo al valore di 0 logico i bit "non significativi", si può vedere che i codici dispositivo per le porte da A a C e per il registro di controllo sono:

PORTA A: 360_8 PORTA C: 362_8
PORTA B: 361_8 REGISTRO DI CONTROLLO: 363_8

Come principio generale, dopo l'interfacciamento del PPI al microcomputer, si dovrà costruire uno schema che indichi, il formato del codice di indirizzo di selezione dispositivo del circuito. Un tale schema è estremamente utile, poichè esso è caratteristico del circuito del PPI e può essere usato per ottenere i giusti codici d'indirizzo per la configurazione del PPI e l'accesso alle sue porte, per tutto il tempo in cui i circuiti del PPI rimangono invariati.

3-3. PROGRAMMAZIONE

Una volta determinata la parola di controllo modo, che configurera le porte del PPI per il modo 0 in ingresso, o in uscita, conoscendo il codice del dispositivo per le porte e per il registro di controllo, la programmazione necessaria diventa poi cosa relativamente semplice. La Fig. 3-3 mostra un flowchart (diagramma di flusso) di un semplice programma, il quale inserisce dati dalla porta B di Fig. 3-1 e manda fuori i risultati dell'elaborazione sul display a LED sulla porta A.

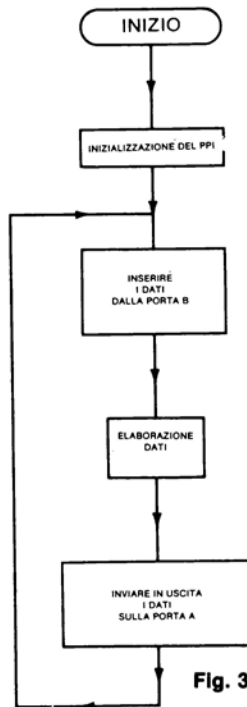



Fig. 3-3. Flowchart per l'I/O (semplice) in modo 0.

Un programma tipico di I/O semplice, per tale flowchart è il Programma 3-1.

Programma 3-1 (Fig. 3-4)

Notare come sono stati usati, in questo programma, la parola di controllo modo e il codice di dispositivo del registro di controllo e come sono stati usati i codici dispositivo, delle porte A e B rispettivamente per l'uscita e l'ingresso.

```
                  /ESEMPIO DI PROGRAMMA PER SEMPLICE I/O IN MODO 0
                  /
                  DB MODE 202
                  DB PORTA 360
                  DB PORTB 361
                  DB CNTRL 363
                  *000 000
000 000 076             MVIA      /CARICARE IN A
000 001 202             MODE      /IL BYTE DI CONTROLLO DEL MODO
000 002 323             OUT       /INVIARE LA PAROLA DI CONTROLLO DEL

000 003 363             CNTRL     /REGISTRO DI CONTROLLO
000 004 333       LOOP, IN        /INSERIRE DATI DALLA PORTA B
000 005 361             PORTB     /CODICE DISPOSITIVO DELLA PORTA B
                  /     .
                  /     .
                  /     - ELABORAZIONE DATI -
                  /     .
                  /     .
                  *000 070
000 070 323             OUT       /INVIARE IL CONTENUTO
                                  /DELL'ACCUMULATORE
000 071 360             PORTA     /ALLA PORTA A
000 072 303             IMP       /SALTARE ALL'INSERZIONE DI NUOVI
                                  /DATI
000 073 004             LOOP      /L0 BYTE INDIRIZZO
000 074 000             0         /HI BYTE INDIRIZZO
```

Fig. 3-4. Programma 3-1.

3-4 DIAGRAMMA DELLA TEMPORIZZAZIONE

La Fig. 3-5, mostra un diagramma di temporizzazione tipico per il modo 0 in ingresso ed uscita. Il diagramma riporta le variazioni logiche di $\overline{RD}$, $\overline{WR}$ e degli ingressi D0-D7 del PPI, durante l'esecuzione dei passi di programma fra le locazioni di memoria $000_8 004_8$ e $000_8 071_8$ in modo 0, per il programma di I/O semplice riportato in Fig. 3-4.

Il punto importante da notare con l'I/O semplice, è che *il trasferimento dati avviene immediatamente in seguito alla ricezione di un segnale logico basso sugli ingressi* $\overline{RD}$ *o* $\overline{WR}$ *del PPI*, in funzione del fatto che i dati siano stabiliti o no e in funzione della prontezza delle periferiche. Per l'ingresso i dati vengono passati, dalla porta indirizzata in quel momento, attraverso il PPI al microcomputer, non appena la linea $\overline{RD}$ va al valore logico basso. Allo stesso modo, un valore logico basso sulla

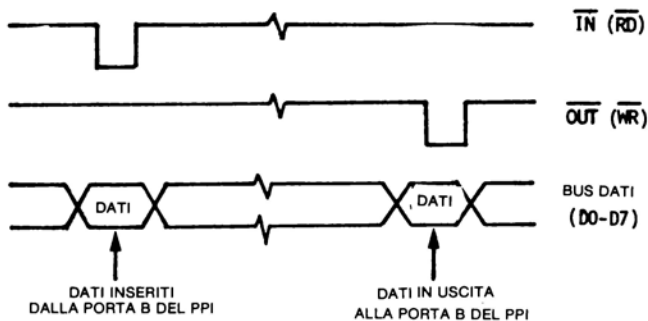



Fig. 3-5. Diagramma di temporizzazione per semplice I/O in modo 0.

linea $\overline{WR}$, causerà il *latch* (*cattura*) *immediato* dei dati presenti sul bus dati del microcomputer, da parte della porta del PPI indirizzata in quel momento. Per l'I/O in accumulatore essa è, di certo, il contenuto del registro A (accumulatore).

3-5. SOTTOPORTE A 4 BIT DELLA PORTA C

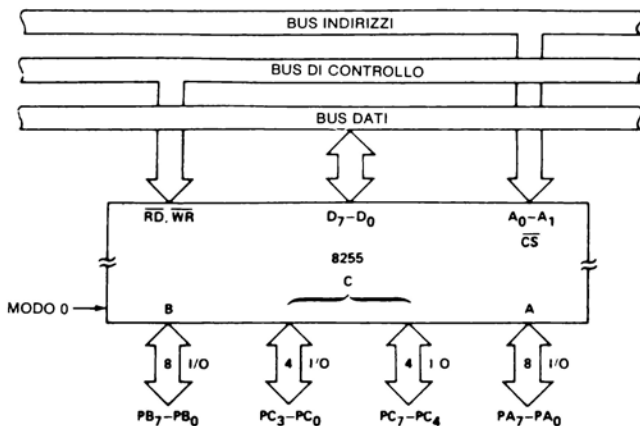
In un paragrafo precedente di questo capitolo, si è visto che i bit da D0 a D3 della parola di controllo del modo, potevano essere usati per assegnare separatamente, rispettivamente la porta C superiore (bit PC7-PC4) e la porta C inferiore (bit PC3-PC0), per il modo 0 in ingresso od in uscita (Fig. 2-2A). Quindi la porta C, nell'I/O in modo 0, può essere effettivamente divisa in due sottoporte a 4 bit, come indicato in Fig. 3-6.

Per gentile concessione dell'Intel Corp.

Fig. 3-6. Schema dei gruppi di linee d'interfaccia disponibili con l'8255 in semplice I/O in modo 0.

Mentre gli 8 bit della porta C possono essere impiegati come due porte I/O a 4 bit, a tali sottoporte *non si può accedere indipendentemente*. Le comunicazioni dati al e dal PPI, avvengono per mezzo di byte di 8 bit. Quindi, durante l'uso di entrambe le porte C, superiore e inferiore, bisogna fare ogni volta attenzione per assicurarsi che il contenuto di una sottoporta, non venga accidentalmente alterato dalla scrittura di dati nell'altra sottoporta. La tabella 3-1, indica le sei condizioni permesse per accedere alle due sottoporte della porta C.

Tabella 3-1. Assegnazione delle sottoporte I/O della porta C

Compiti	Porta C superiore	Porta C inferiore	Requisiti
Ingresso	Ingresso	Uscita	Mascherare i 4 bit non significativi
Ingresso	Uscita	Ingresso	
Ingresso	Ingresso	Ingresso	
Uscita	Ingresso	Uscita	Dati posti nei 4 bit inferiori
Uscita	Uscita	Ingresso	Dati posti nei 4 bit superiori
Uscita	Uscita	Uscita	Dati posti nei 4 bit superiori e inferiori

Per l'ingresso dati da una sottoporta della porta C, viene impiegata una tecnica di mascheratura che permette di ricavare i quattro bit richiesti. Così, dopo aver determinato da quale sottoporta si vuole il dato, si usa un'istruzione di ingresso per inserire gli 8 bit della porta C. Mediante una maschera ad 8 bit, si eliminano poi i 4 bit non voluti. Se interessano i quattro bit della porta C superiore, possono essere necessarie quattro istruzioni RAR (rotazione a destra con carry) per far scorrere tali bit nelle quattro posizioni meno significative, se, ad esempio, i 4 bit rappresentano un numero BCD, oppure i quattro bit meno significativi di un contatore di eventi. Il programma 3-2, mostra i passi del programma necessari per inserire i dati dalla porta C superiore impiegando il circuito di Fig. 3-1. Notare che la parola di controllo del modo per l'inizializzazione del PPI, è stata cambiata con il valore 212. Il bit D3 della parola di controllo del modo, è stato cambiato da zero ad uno per ridefinire la porta C superiore come ingresso.

Programma 3-2. Lettura dalla porta C superiore (Fig. 3-7)

Per l'uscita di dati da entrambe le porte, C superiore ed inferiore, si pongono semplicemente i dati nei corrispondenti 4 bit dell'accumulatore e si invia quindi il contenuto di tale registro alla porta C. Il contenuto degli altri quattro bit dell'accumulatore (cioè, i quattro bit superiori se si

```
                    /
                    /PROGRAMMA D'INGRESSO ALLA PORTA C
                    /
                    DB MODE 212
                    DB PORTC 362
                    DB CNTRL 363
                   *001 000
001 000 076                 MVIA        /CARICARE IN A LA PAROLA DI CONTROLLO
001 001 212                 MODE        /DEL MODO CHE SEGUE
001 002 323                 OUT         /INVIARE LA PAROLA DI CONTROLLO DEL
                                        /MODO AL
001 003 363                 CNTRL       /REGISTRO DI CONTROLLO, PER
                                        /L'INIZIALIZZAZIONE DEL PPI
001 004 333                 IN          /INSERIRE I DATI DALLA PORTA C
001 005 362                 PORTC       /CODICE DISPOSITIVO DELLA PORTA C
001 006 346                 ANI         /MASCHERARE I 4 BIT SUPERIORI
001 007 360                 360         /BYTE MASCHERA
001 010 037                 RAR         /ROTAZIONE A DESTRA CON CARRY
001 011 037                 RAR         /ROTAZIONE A DESTRA CON CARRY
001 012 037                 RAR         /ROTAZIONE A DESTRA CON CARRY
001 013 037                 RAR         /ROTAZIONE A DESTRA CON CARRY
001 014 166                 HLT         /HALT
```

Fig. 3-7. Programma 3-2.

sta inviando un dato alla porta C inferiore) non è importante, *purchè l'altra sottoporta sia programmata come ingresso*. Ad ogni modo se anche l'altra sottoporta è programmata come uscita, bisogna fare attenzione per evitare che i 4 bit associati a tale sottoporta vengano accidentalmente cambiati da un'operazione d'uscita sulla metà voluta della porta C. Ciò, è fatto assicurandosi che il byte inviato alla porta C sia composto dai nuovi quattro bit, per la sottoporta che deve essere aggiornata, insieme ai 4 precedentemente inviati all'altra sottoporta. Detti valori dovranno probabilmente essere salvati in un registro general-purpose, o in una

Programma 3-3. Scrittura della porta C inferiore (Fig. 3-8)

```
                    DB PORTC 362
                    *020 000
                    /       .
                    /       .
                    /       .
020 000 006                 MVIB        /CARICARE I DATI NEL REGISTRO B
020 001 010                 010         /PER LA PORTA C INFERIORE
020 002 016                 MVIC        /CARICARE I DATI NEL REGISTRO C
020 003 200                 200         /PER LA PORTA C SUPERIORE
020 004 171                 MOVAC       /INVIARE I DATI DELLA PORTA C
                                        /SUPERIORE IN A
020 005 346                 ANI         /AZZERARE I QUATTRO BIT INFERIORI
020 006 360                 360
020 007 260                 ORAB        /OR CON I DATI DELLA PORTA C SUPERIORE
020 010 323                 OUT         /INVIARE IL BYTE DATI IN USCITA ALLA
020 011 362                 PORTC       /PORTA C
                    /       .
                    /       .
                    /       .
```

Fig. 3-8. Programma 3-3.

locazione di memoria di lettura/scrittura. Il programma 3-3 (Fig. 3-8), illustra la tecnica per l'uscita di dati dalla porta C inferiore, quando si hanno dati in uscita anche sulla porta C superiore. Si immagina che il PPI sia già stato inizializzato con entrambe le porte C superiore ed inferiore, configurate come uscite.

3-6 BIBLIOGRAFIA

[1] Osborne Adam *Un'Introduzione ai Microcomputer*, vol. 1 - Il libro dei concetti fondamentali, Gruppo Editoriale Jackson, Milano, 1980, Adam Osborne and Associates, Inc., Berkeley, CA, 1976.

[2] Hilburn, J. L. & Julich, P. M. *Microcomputers/Microprocessors: Hardware, Software and Applications,* Prentice-Hall, Englewood Cliffs, 1976.

[3] Soucek, B. *Microprocessors & Microcomputers*, J. Wiley & Sons, New York, 1976.

SCHEMA A BLOCCHI DELL'8255

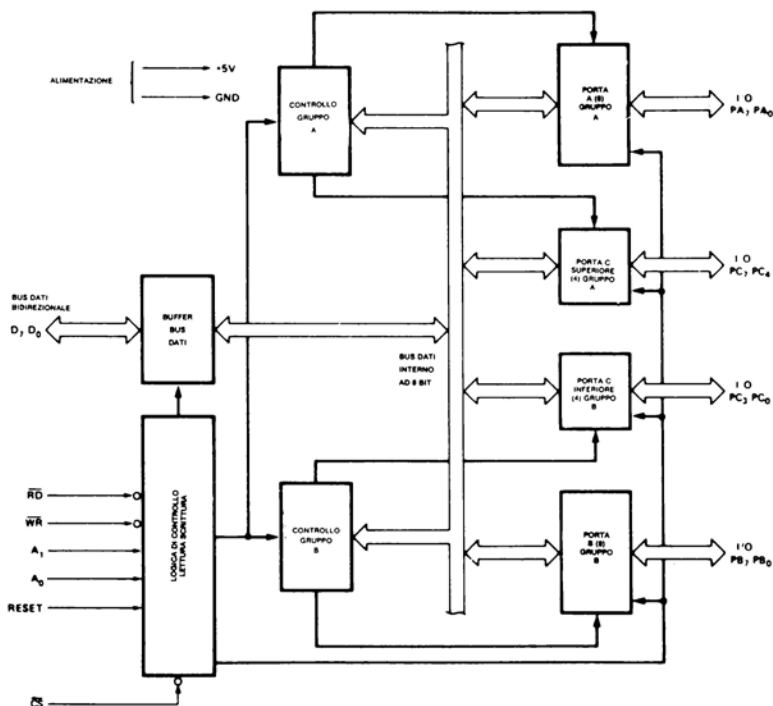
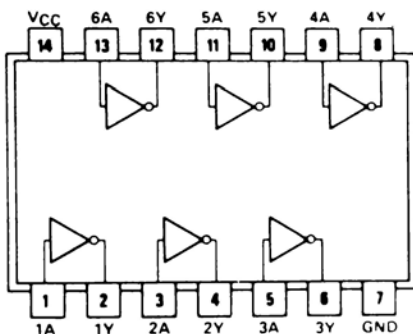



Fig. 3-9. Schema a blocchi e configurazione dei pin dell'IC.

3-7 SOMMARIO DEGLI ESPERIMENTI DA 3-1 A 3-3

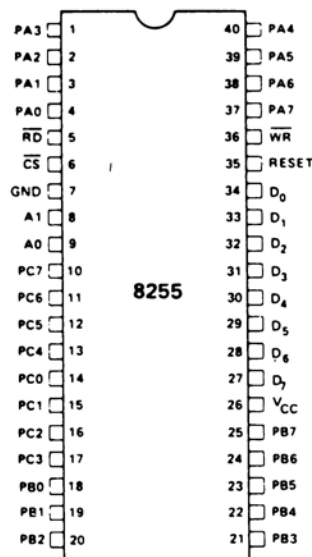
Esperimento	*Descrizione*
3-1	(a) Interfacciare il PPI 8255 per semplice I/O in accumulatore. (b) Illustrare l'impiego del PPI 8255 nel *funzionamento in modo 0* per *l'uscita dati.*
3-2	Illustrare l'impiego del PPI 8255 nel *funzionamento in modo 0* per *l'ingresso dati.*
3-3	Lo scopo di questo esperimento è quello di illustrare il funzionamento dell'8255 in modo 0 per operazioni combinate di ingresso e uscita. Tale esperimento rappresenta una sintesi delle procedure illustrate individualmente negli Esperimenti 3-1 e 3-2 e può essere quindi considerato come una verifica della comprensione, hardware e software, del funzionamento del PPI in modo 0.

Configurazione del pin e schema a blocchi del circuito integrato (Fig. 3-9)

NOMI DEI PIN

D_7–D_0	BUS DATI (BIDIREZIONALE)
RESET	INGRESSO DI RESET
$\overline{CS}$	SELEZIONE CHIP
$\overline{RD}$	INGRESSO LETTURA
$\overline{WR}$	INGRESSO SCRITTURA
A0, A1	INDIRIZZO PORTA
PA7-PA0	PORTA A (BIT)
PB7-PB0	PORTA B (BIT)
PC7-PC0	PORTA C (BIT)
V_{CC}	+5 VOLTS
GND	Ø VOLTS

CONFIGURAZIONE DEI PIN

8255

Segnale	Pin	Pin	Segnale
PA3	1	40	PA4
PA2	2	39	PA5
PA1	3	38	PA6
PA0	4	37	PA7
$\overline{RD}$	5	36	$\overline{WR}$
$\overline{CS}$	6	35	RESET
GND	7	34	D_0
A1	8	33	D_1
A0	9	32	D_2
PC7	10	31	D_3
PC6	11	30	D_4
PC5	12	29	D_5
PC4	13	28	D_6
PC0	14	27	D_7
PC1	15	26	V_{CC}
PC2	16	25	PB7
PC3	17	24	PB6
PB0	18	23	PB5
PB1	19	22	PB4
PB2	20	21	PB3

ESPERIMENTO 3-1
OPERAZIONI USCITA DATI IN MODO 0

Scopo

I propositi di questo esperimento sono i seguenti:

(a) Interfacciare il chip del circuito integrato 8255 per il semplice funzionamento di I/O in accumulatore.

Programma (Fig. 3-10)

```
                /
                /PROGRAMMA D'USCITA IN MODO 0 DEL PPI
                /
                DB MODE 200
                DB DATA 200
                DB CNTRL 203
                * 003 000
003 000 061             LXISP.      /CARICARE LO STACK POINTER CON
003 001 200             200         /LO BYTE INDIRIZZO
003 002 003             003         /HI BYTE INDIRIZZO
003 003 076             MVIA        /METTERE IN ACCUMULATORE IL
003 004 200             MODE        /BYTE DI CONTROLLO DEL MODO 0
003 005 323             OUT         /INVIARE IL CONTENUTO DI A AL
003 006 203             CNTRL       /REGISTRO DI CONTROLLO
003 007 323     LOOP,   OUT         /INVIARE IL CONTENUTO DI A ALLA
003 010 200             DATA        /PORTA A
003 011 074             INRA        /INCREMENTARE A
003 012 315             CALL        /CHIAMATA ALLA SUBROUTINE "DELAY"
003 013 200             DELAY
003 014 003             0
003 015 303             JMP         /SALTO INCONDIZIONATO A "LOOP"
003 016 007             LOOP
003 017 003             0
                /
                /SUBROUTINE: DELAY
                /DESCRIZIONE: QUESTA SUBROUTINE GENERA
                /UN RITARDO DI 0,1 SECONDI PER UN MICROCOMPUTER
                /FUNZIONANTE CON UN CLOCK DI 750 KHZ
                /
                * 003 200
003 200 365     DELAY,  PUSHPSW     /SALVARE LO STATO DEL MICROCOMPUTER
003 201 325             PUSHD
003 202 021             LXID        /CARICARE NELLA COPPIA DI REGISTRI D I
003 203 250             250         /BYTE DI TEMPORIZZAZIONE
003 204 015             015         /PER UN RITARDO DI 0,1 SECONDI
003 205 033     LOOP1,  DCXD
003 206 173             MOVAE
003 207 262             ORAD        /IL BYTE DI TEMPORIZZAZIONE È ZERO?
003 210 302             JNZ         /NO, CONTINUARE IL DECREMENTO
003 211 205             LOOP1
003 212 003             O
003 213 321             POPD        /SI', RIPRISTINARE LO STATO DEL
                                    /MICROCOMPUTER
003 214 361             POPPSW
003 215 311             RET
```

Fig. 3-10. Programma per l'Esperimento 3-1.

(b) Illustrare l'impiego dell'8255 nel funzionamento in modo 0 per l'uscita dati.

Passo 1

Collegare il circuito indicato nello schema di Fig. 3-11. Gli ingressi A0, A1 e A7, sono disponibili sul bus indirizzi della memoria.

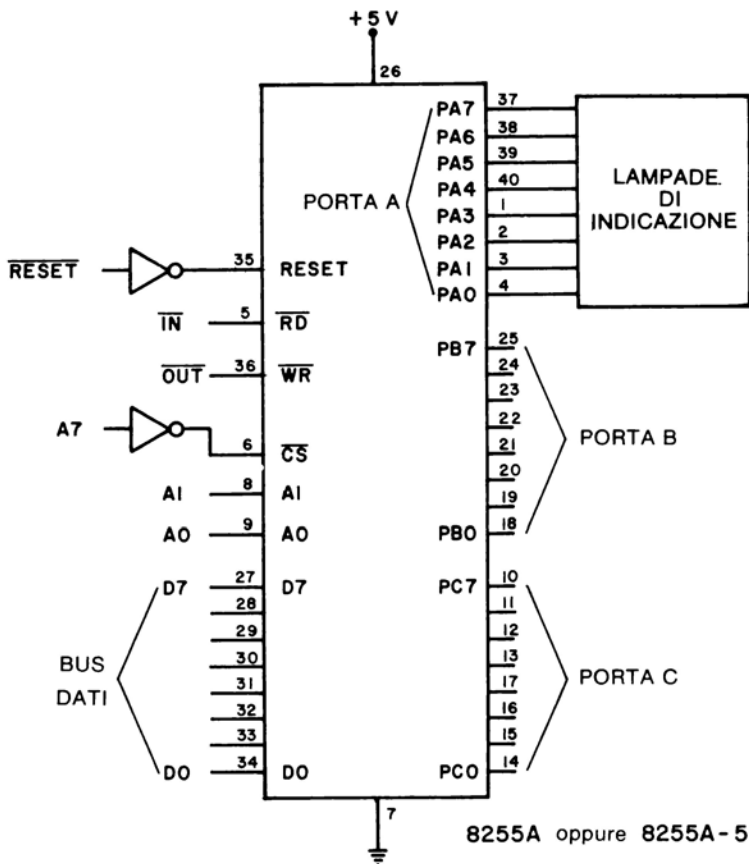



Fig. 3-11. Impiego dell'IC 8255.

Dal momento che, sulle linee indirizzi da A0 ad A7 e da A8 ad A15 durante un'istruzione di OUT o IN, compare lo stesso set di codici di dispositivo, i bit A7 e A15 sono, in questo caso, identici. Fare attenzione che sia fatto il giusto collegamento, sull'ingresso RESET al pin 25 del chip 8255. L'autore una volta, effettuò un collegamento di RESET

difettoso e spese due ore cercando di risolvere il problema, che egli pensava essere altrove nel circuito.

Passo 2

La tabella della verità per il funzionamento del chip interfaccia periferica programmabile 8255, è riportata in Tabella 3-2.

Tabella 3-2. Operazioni di Base dell'IC 8255

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	**Operazione d'Ingresso (Lettura)**
0	0	0	1	0	Porta A → Bus Dati
0	1	0	1	0	Porta B → Bus Dati
1	0	0	1	0	Porta C → Bus Dati
					Operazione d'Uscita (Scrittura)
0	0	1	0	0	Bus Dati → Porta A
0	1	1	0	0	Bus Dati → Porta B
1	0	1	0	0	Bus Dati → Porta C
1	1	1	0	0	Bus Dati → Controllo
					Disabilitazione della funzione
X	X	X	X	1	Bus Dati → 3-State
1	1	0	1	0	Condizione non ammessa

Schema del Circuito (Fig. 3-11)

Studiare la tabella della verità in Tabella 3-3 per i quattro ingressi di controllo del chip 8255. Supporre che l'ingresso $\overline{CS}$ sia al valore logico 0, cioè, che A7 sia al valore logico 1.
Ricordarsi che il bus indirizzi, quando usato per indirizzare il chip 8255, ha il seguente significato:

Bit:	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
Ingresso:	$\overline{CS}$	**X**	**X**	**X**	**X**	**X**	**A1**	**A0**

Un X indica che lo stato logico, di quel bit d'indirizzo, non è significativo. Quale valore d'indirizzo del dispositivo è necessario per scrivere un byte di controllo nel registro di controllo?

Per scrivere nel registro di controllo, sia A7 che A1 ed A0 devono essere al valore logico 1, in accordo con la tabella della verità. Quindi il codice del dispositivo è:

Tabella 3-3. Ingressi di controllo dell'IC 8255

OUT	IN	A1	A0	Azione
0	1	0	0	Bus Dati → Porta A
0	1	0	1	Bus Dati → Porta B
0	1	1	0	Bus Dati → Porta C
0	1	1	1	Bus Dati → Controllo Register
1	0	0	0	Porta A → Bus Dati
1	0	0	1	Porta B → Bus Dati
1	0	1	0	Porta C → Bus Dati
1	0	1	1	Non consentito
1	1	0	0	Bus Dati → 3-State
1	1	0	1	Bus Dati → 3-State
1	1	1	0	Bus Dati → 3-State
1	1	1	1	Bus Dati → 3-State
0	0	X	X	Non consentito

1XXXXX11

Se si considera X uguale al valore logico 0, l'indirizzo del dispositivo è semplicemente 10000011, ovvero 203. Questa è la tecnica, da noi usata, per indirizzare il chip 8255.

Passo 3

La configurazione del modo 0, che si controllerà per prima, è per la parola di controllo del modo 0, che è data dalla Intel Corporation come in Fig. 3-12. Quello è il valore della parola di controllo ad 8 bit, MODE, che deve comparire nel programma, nel L0 indirizzo di memoria 004.

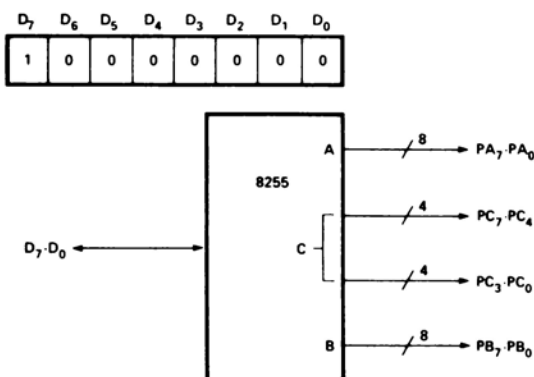



Fig. 3-12. Parola di controllo e schema per il modo 0.

Lo schema indica che si faranno uscire i dati della porta A del chip 8255. L'indirizzo del dispositivo per la porta A, è semplicemente 200, in accordo con la tabella della verità fornita nel Passo 2. Tale codice del dispositivo, DATA, deve comparire nel programma all'indirizzo L0 di memoria 010.

Passo 4

Caricare il programma in memoria. Fare attenzione che la parola di controllo del modo e il codice del dispositivo I/O, PPI, siano corretti.

Passo 5

Eseguire il programma. Che cosa si vede sulla coppia di monitor Outboards™ * collegati alla porta A?

Noi vediamo che i monitor vengono incrementati alla frequenza di circa 10Hz.

Passo 6

Trasferire la coppia di monitor Outboards alla porta B. Cambiare l'indirizzo del dispositivo I/O, PPI, nel L0 indirizzo di memoria 010 con 201. Eseguire il programma. Cosa si osserva?

Noi vediamo ancora che i monitor vengono incrementati alla frequenza di circa 10Hz.

Passo 7

Trasferire la coppia di monitor Outboards alla porta C. Cambiare l'indirizzo del dispositivo nel L0 indirizzo di memoria 010 con 202. Eseguire il programma. Che cosa si osserva ora?

---

* OUTBOARD è un marchio registrato della E & L Instruments, Inc.

Noi vediamo ancora che i monitor alla porta C, vengono incrementati alla frequenza di circa 10Hz.

Per riassumere, i valori del programma che si sono usati, sono i seguenti:

Parola di controllo	: 200	Stabilisce il funzionamento del PPI in modo 0 e predispone le porte da A a C come uscite.
Porta A (uscita)	: 200	Codici del dispositivo I/O
Porta B (uscita)	: 201	rispettivamente per le
Porta C (uscita)	: 202	porte da A e C.

Passo 8

Con la coppia di monitor Outboards, ancora assegnati alla porta C, cambiare la parola di controllo del modo, MODE, all'indirizzo di memoria 004, con quella indicata in Fig. 3-13. Eseguire il programma. Che cosa si osserva sulla coppia di monitor Outboards? Spiegare ciò che si vede.

Noi vediamo che vengono incrementati solamente i LED collegati ai pin inferiori della porta C (PC0-PC3). Ciò a causa del fatto che, la parola di controllo del modo 210, assegna la porta C inferiore come uscita, ma configura la porta C superiore come ingresso. Quindi, sebbene si inviino 8 bit in uscita alla porta C, solamente i quattro inferiori vengono visualizzati.

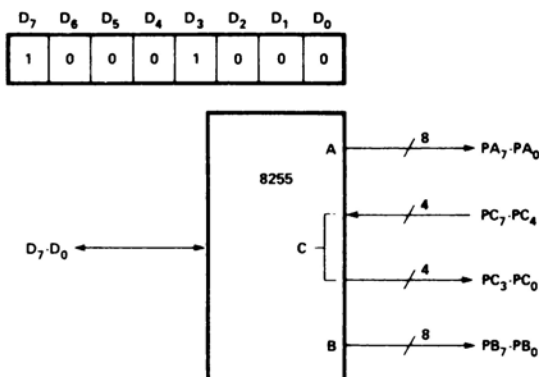



Fig. 3-13. Parola di controllo del secondo modo.

Passo 9

Con la coppia di monitor Outboards, ancora assegnati alla porta C, cambiamo ora "MODE", all'indirizzo di memoria 004, con il valore riportato in Fig. 3-14. Eseguire il programma. Che cosa si osserva sulla coppia di monitor Outboards? Spiegare ciò che si vede.

Noi, questa volta, vediamo che i LED collegati alla porta C superiore, sono lentamente incrementati, mentre quelli collegati alla porta C inferiore non sono luminosi. Ciò a causa del fatto che, la parola di controllo del modo 201, predispone la porta C superiore come uscita e la porta C inferiore come ingresso. Solamente i quattro bit superiori del byte di 8 bit, che si è inviato alla porta C, vengono quindi visualizzati.

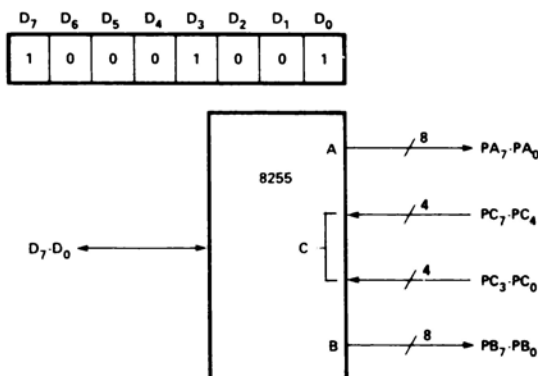



Fig. 3-14. Parola di controllo del terzo modo.

Passo 10

Con la coppia di monitor Outboards, ancora assegnati alla porta C, cambiare la parola di controllo, nel L0 indirizzo di memoria 004, con quella riportata in Fig. 3-15. Eseguire il programma corretto e spiegare che cosa si vede sulla coppia di monitor Outboards.

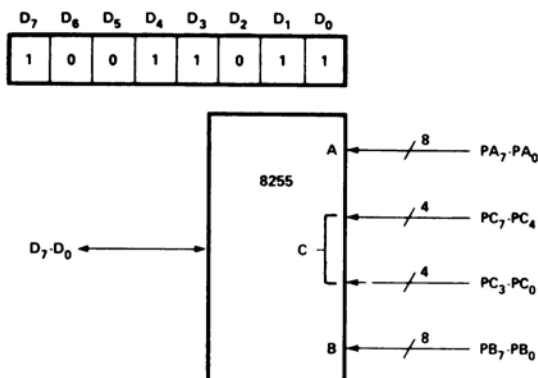



Fig. 3-15. Parola di controllo del quarto modo.

Questa volta nessun LED è luminoso durante l'esecuzione del programma, poichè la parola di controllo del modo 211, predispone sia la porta C superiore che la porta C inferiore, come ingressi. Quindi nessuno degli 8 bit del byte inviato alla porta C dall'accumulatore, viene visualizzato.

NOTA: Rimuovere i monitor Outboards che sono collegati alla porta C. Non fare alcun altro cambiamento all'hardware se si vuole eseguire il prossimo esperimento.

Domande

1. Quale codice del dispositivo e quale comando d'ingresso/uscita (cioè, IN o OUT) sono necessari, se si vogliono *leggere i dati* che compaiono *alla porta A*? Porre tutte le X al valore logico 0.

2. Quale codice del dispositivo e comando di ingresso/uscita sono necessari se si vogliono *far uscire dei dati dalla porta B*? Porre tutte le X al valore logico 0.

3. Quale codice del dispositivo e comando di ingresso/uscita sono necessari per *far uscire dati dalla porta C*? Porre tutte le X al valore logico 0.

4. Quanti codici del dispositivo si devono impiegare per le operazioni di ingresso uscita sul chip 8255? Si può supporre che ogni X possa essere sia al valore logico 0 che al valore logico 1.

5. Indicare almeno un modo per diminuire il numero dei possibili codici del dispositivo impiegati, come risulta dal calcolo nella precedente domanda N° 4. Se necessario disegnare uno schema del circuito.

6. Quali segnali di controllo e quale codice del dispositivo permettono di leggere lo stato degli 8 bit del registro di controllo?

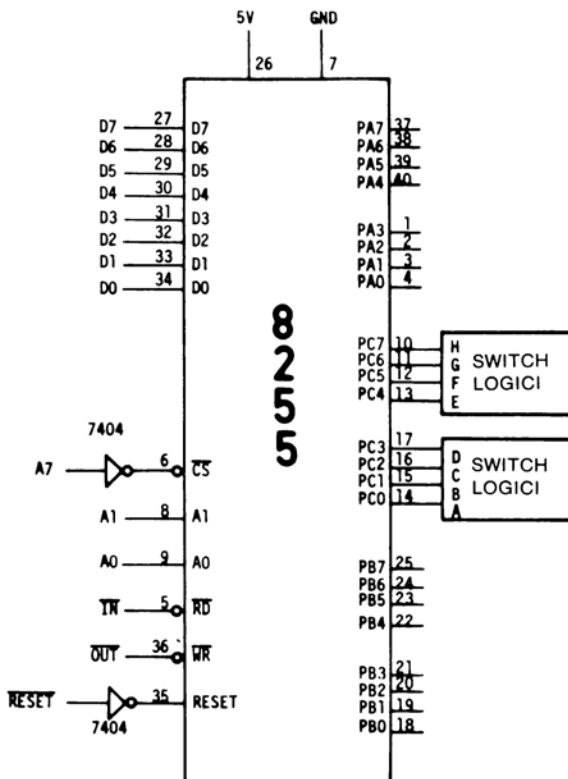



Fig. 3-16. Circuito per l'Esperimento 3-2.

ESPERIMENTO 3-2
OPERAZIONI INGRESSO DATI IN MODO 0

Scopo

Lo scopo di questo esperimento è di illustrare il *funzionamento in modo 0* dell'8255 per *l'ingresso di dati nell'accumulatore*, attraverso il chip 8255, da una coppia di switch logici Outboards.

Schema del circuito (Fig. 3-16)

NOTA: Questo circuito è lo stesso di quello collegato nell'Esperimento 3-1, ad eccezione dell'impiego di switch logici, al posto dei monitor, alle linee delle porte da A a C.

Programma (Fig. 3-17)

```
                  /
                  /PROGRAMMA D'INGRESSO NEL PPI IN MODO 0
                  /
                  DB MODE 233
                  DB DATA 202
                  DB CNTRL 203
                  *003 000
003 000 076              MVIA      /CARICARE IN A
003 001 233              MODE      /IL BYTE DI CONTROLLO DEL MODO 0
003 002 323              OUT       /INVIARE IL CONTENUTO DI A AL
003 003 203              CNTRL     /REGISTRO DI CONTROLLO
003 004 333       LOOP,  IN        /INSERIRE I DATI DALLA
003 005 202              DATA      /PORTA D'INGRESSO DEL PPI
003 006 323              OUT       /INVIARLI IN USCITA ALLA PORTA 2
003 007 002              002
003 010 303              JMP       /SALTARE A LOOP
003 011 004              LOOP
003 012 003              0
```

Fig. 3-17. Programma per l'Esperimento 3-2.

Passo 1

Collegare il circuito indicato nello schema di Fig. 3-16, se non è già collegato dal precedente esperimento. La configurazione del modo 0 che si vuole controllare per prima, è data dalla parola di controllo del modo 0 che è riportata in Fig. 3-18. Quello è il valore degli 8 bit della parola di controllo, che si devono inserire nel L0 indirizzo di memoria 001 nel programma.

Lo schema di Fig. 3-18, indica che si vogliono inserire i dati attraverso la porta C dell'8255. Il codice del dispositivo per la porta C, è 202, in accordo con la tabella della verità fornita nell'Esperimento 3-1. Tale codice deve comparire al L0 indirizzo di memoria 005 nel programma.

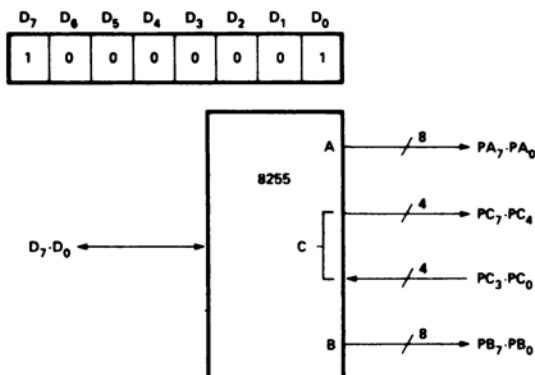



Fig. 3-18. Parola di controllo e schema per il modo 0.

Passo 2

Caricare il programma nella memoria di lettura/scrittura. Accertarsi che la parola di controllo del modo ed il codice dispositivo ingresso/uscita del PPI, siano corretti.

Passo 3

Eseguire il programma. Mentre il programma viene eseguito, variare lo switch logico montato e osservare alla porta d'uscita 002 i byte che vengono inseriti. Cosa accade?

Noi vediamo che il byte di 8 bit, che è impostato sugli switch logici, viene visualizzato continuamente alla porta d'uscita 002. Ogni variazione sugli switch logici, produce una corrispondente immediata variazione alla porta d'uscita 002.

Passo 4

Spostare la coppia di switch logici Outboards alla porta B. Cambiare l'indirizzo del dispositivo al L0 indirizzo di memoria 005 con 201. Eseguire il programma. Cosa accade ora?

Noi vediamo che il byte visualizzato alla porta 2, è ancora lo stesso impostato alla porta B del PPI, dagli switch logici.

Passo 5

Infine, spostare la coppia di switch logici Outboards alla porta A. Cambiare il codice del dispositivo al L0 indirizzo di memoria 005 con 200. Eseguire il programma. Si osserva ancora che i byte inseriti dagli switch logici alla porta A, vengono visualizzati alla porta d'uscita 002 quando si fa eseguire il programma? Si deve vedere che accade ciò.

ESPERIMENTO 3-3
OPERAZIONI COMBINATE INGRESSO ED USCITA IN MODO 0

Scopo

Lo scopo di questo esperimento è di illustrare il funzionamento del chip 8255 per ingresso ed uscita combinate, impiegando la porta C come ingresso e la porta B come uscita. Anche la porta A, sarà programmata

Schema del circuito (Fig. 3-19)

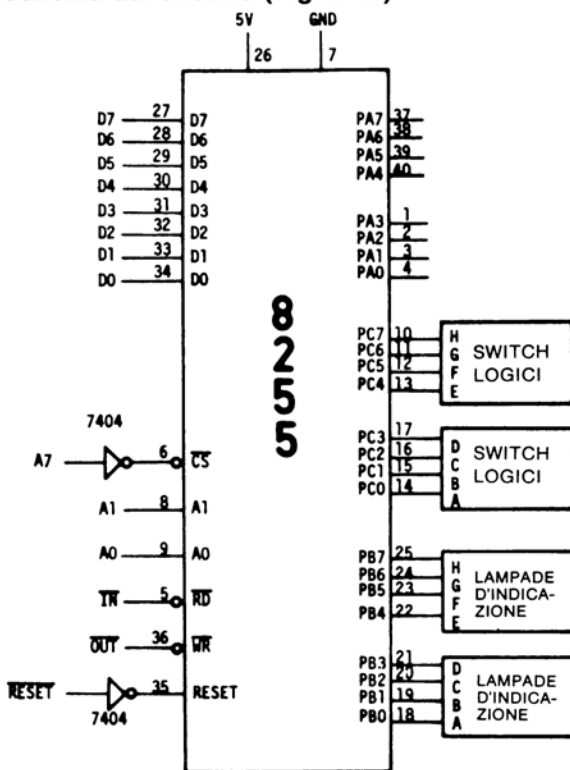



Fig. 3-19. Circuito per l'Esperimento 3-3.

Programma (Fig. 3-20)

```
                /
                /PROGRAMMA INGRESSO E USCITA COMBINATI NEL PPI IN
                /MODO 0
                /
                DB MODE 232
                DB CNTRL 203
                DB DATA1 202
                DB DATA2 201
                *003 000
003 000 076            MVIA     /CARICARE IN A
003 001 232            MODE     /LA PAROLA DI CONTROLLO DEL MODO 0
003 002 323            OUT      /INVIARE IL CONTENUTO DI A AL
003 003 203            CNTRL    /REGISTRO DI CONTROLLO
003 004 333     LOOP,  IN       /CARICARE IN A IL CONTENUTO
003 005 202            DATA1    /DELLA PORTA D'INGRESSO DEL PPI
003 006 000            NOP
003 007 000            NOP
003 010 323            OUT      /INVIARE IL CONTENUTO DI A ALLA
003 011 201            DATA2    /PORTA D'USCITA DEL PPI
003 012 303            JMP      /SALTARE A "LOOP"
003 013 004            LOOP
003 014 003            0
```

Fig. 3-20. Programma per l'Esperimento 3-3.

come uscita. Questo esperimento è una sintesi delle procedure illustrate individualmente, rispettivamente negli Esperimenti 3-1 e 3-2, per l'uscita e l'ingresso di dati. Se credete di aver compreso il procedimento necessario per il funzionamento ingresso/uscita in modo 0 del PPI, potete probabilmente scrivere e controllare un programma per inserire dati dagli switch logici alla porta C e farli uscire sui monitor alla porta B (o porta A). L'autore ha scritto un programma, se preferite non far tentativo in questo stadio. La decisione su questo esperimento spetta a voi - divertitevi!

Passo 1

La configurazione del modo 0 che si controllerà per prima, è definita dalla parola di controllo del modo di Fig. 3-21. Questo è il valore degli 8 bit della parola di controllo, MODE, che compare al L0 indirizzo di memoria, nel programma di Fig. 3-20. Lo schema di Fig. 3-21, indica che si inseriranno dati tramite la porta C, la quale ha il codice del dispositivo 202, mentre saranno fatti uscire dalla porta B, la quale ha il codice del dispositivo 201. Tali due indirizzi dovranno comparire rispettivamente ai L0 indirizzi di memoria 005 e 001.

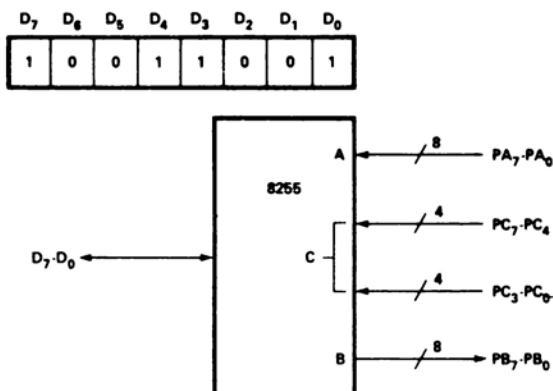



Fig. 3-21. Parola di controllo del modo e schema.

Passo 2

Caricare il programma nella memoria di lettura/scrittura. Accertarsi che la parola di controllo ed i due indirizzi del dispositivo, siano giusti.

Passo 3

Eseguire il programma. Mentre il microcomputer sta lavorando, variare gli switch logici e osservare l'uscita alla porta B. Cosa accade?

Noi osserviamo, che il byte di dati che è impostato sugli switch logici alla porta C, è visualizzato continuamente alla porta B e viene aggiornato immediatamente quando gli switch vengono alterati.

Passo 4

Spostare la coppia di monitor alla porta A. Cambiare l'indirizzo del dispositivo al L0 indirizzo di memoria 011 con 200, che è l'indirizzo della porta A; cambiare anche la parola di controllo del modo al L0 indirizzo di memoria 001 con 213, la quale configura il PPI come mostrato in Fig. 3-22. Eseguire il programma ancora una volta anche lo switch logico collegato. Cosa accade ora?

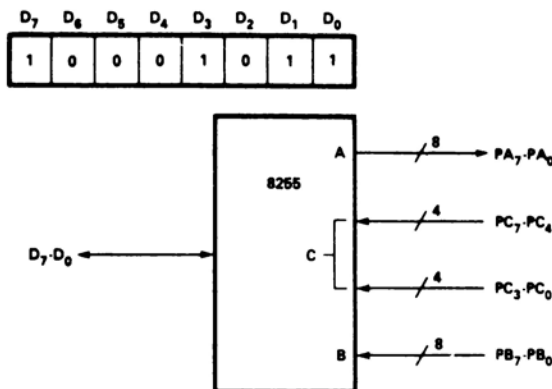



Fig. 3-22. Parola di controllo e schema del secondo modo.

Questa volta il byte di dati impostato dagli switch logici alla porta C, viene continuamente visualizzato alla porta A ed è ancora aggiornato immediatamente quando gli switch logici sono riposizionati.

Passo 5

Aggiungere le seguenti istruzioni al programma, al posto delle due istruzioni di non-operazione (NOP) a 003 006 e 003 007:

003 006	346	ANI
003 007	272	272

Fare ora eseguire il programma e commutare, a turno, la posizione di ogni switch, passando dal valore logico 0 ad 1. Cosa accade sulla porta A?

Noi osserviamo che le lampade corrispondenti ai bit PC6, PC2 e PC0, non si accendono. Perchè?

I bit di queste posizioni del byte con cui si fa l'AND in modo immediato sono zero. Qualsiasi valore messo in AND con 0, dà come risultato 0.

Passo 6

Sostituire ora l'istruzione alla locazione di memoria 003 006 con la seguente istruzione:

003 006 356 XRI

Eseguire il programma e spiegare cosa si osserva alla porta A quando gli switch logici vengono, a turno, commutati dal valore logico 0 a 1.

Noi osserviamo che un LED alla porta A si illuminerà, quando lo stato logico di un bit alla porta C sarà quello giusto, per il corrispondente bit nel byte immediato 272, alla locazione 003 007. Ciò poichè la funzione logica di OR esclusivo, è un rilevatore diverso e genera un 1 logico come risultato, quando i due bit messi in OR esclusivo sono diversi.

Passo 7, 8, ..., ecc.

Vi invitiamo ora, a provare ad inserire da soli, a turno, le seguenti istruzioni alla locazione 003 006.

366 ORI
306 ADI
326 SUI

CAPITOLO 4

OPERAZIONE DI SET/RESET BIT SUL PPI

4-1. INTRODUZIONE

La caratteristica dell'8255, di poter eseguire operazioni di set/reset bit, è stata brevemente descritta nel Paragrafo 2-1 (D) del Capitolo 2. Si è detto che questa è essenzialmente una caratteristica della porta C, che permette all'utilizzatore di posizionare individualmente ognuno dei propri bit. Per utilizzare la suddetta possibilità della porta C, bisogna inviare una parola di controllo di set/reset bit, al registro di controllo del PPI. Il formato di questa parola di controllo, lo si è descritto nel Capitolo 2 ed è nuovamente riportato in Fig. 4-1.

Notare che è il bit più significativo (D7) della parola di controllo, che indica se il byte di controllo deve essere interpretato, dall'8255, come una parola di controllo del *modo* (D7=1), o come una parola di controllo *set/reset bit* (D7=0). Altre caratteristiche importanti di tale parola di controllo sono:

- I bit D3, D2 e D1, della parola di controllo set/reset bit, rappresentano un codice a 3 bit il cui valore binario indica il bit della porta C (PC0-PC7) che deve essere posizionato.
- Lo stato logico del bit D0, indica se il bit della porta C selezionato, deve essere posto a 1 (D0=1) o a 0 (D0=0). Ciò significa che ogni parola di controllo di set/reset bit, può essere usata per posizionare solamente uno alla volta dei bit della porta C. Se, ad esempio, il bit PC0 della porta C (D3 D2 D1=000) è stato posto ad 1(D0= 1), inviando una parola di controllo di set/reset bit al registro della parola di controllo del PPI, sarà necessaria una seconda parola di controllo set/reset bit, per azzerare PC0.
- I bit D6, D5 e D4, non vengono usati e solitamente si pongono uguali a 0. In ogni caso, come si vedrà nell'Esperimento 4-1, essi "non sono significativi" e si può quindi usare per essi qualsiasi valore logico.

Avendo parlato della parola di controllo che si invia al registro di controllo del PPI, per posizionare individualmente ogni bit della porta C, è ragionevole chiedersi perchè si sia inserita una tale caratteristica nell'8255. La ragione principale è legata al funzionamento del PPI, nei modi 1 e 2, che sarà descritto nei successivi tre capitoli. Diciamo solamente, per il momento, che i modi di funzionamento 1 e 2, vengono impiegati rispettivamente per handshaking di I/O unidirezionale e bidi-

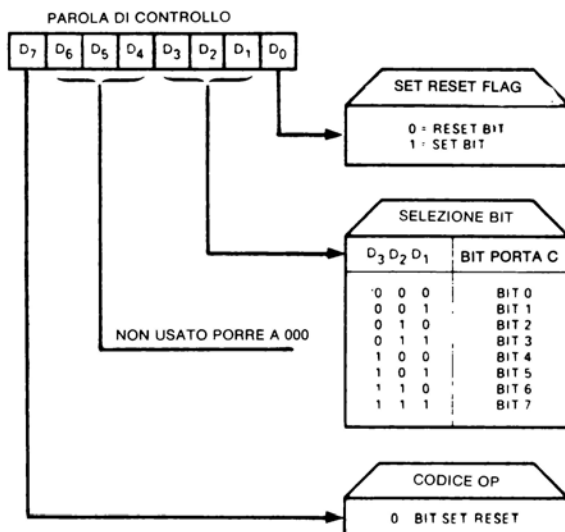



Fig. 4-1. Formato della parola di controllo di set/reset bit.

rezionale, con possibilità di interrupt. I bit della porta C vengono impiegati come bit di controllo di handshaking, per le porte A e B. Tali bit di controllo, sono configurati in modo che i flag della porta A o B, vengano posti ad 1, una volta che una periferica esterna abbia accettato un trasferimento I/O. Associato ad ogni flag di interrupt, c'è un *flip-flop di abilitazione interrupt*, che può essere posizionato impiegando la caratteristica di set/reset bit della porta C. I dettagli di come e perchè ciò sia fatto, saranno spiegati nei Capitoli 6 e 7.

Un secondo impiego più semplice, ma ugualmente importante, della possibilità di set/reset bit della porta C, è di generare livelli logici e impulsi di durata variabile, che possano essere usati come impulsi di gate e di reset hardware per contatori, flip-flop, etc. In generale i bit della porta C, quando usata con la parola di controllo set/reset bit, possono essere impiegati per operazioni di gate, strobe e reset, le quali sono di

solito collegate con la selezione del dispositivo o impulsi di indirizzo di selezione*. Il maggior vantaggio derivante dall'uso dei bit della porta C del PPI, per le suddette funzioni, è, che non è necessario alcun hardware aggiuntivo. Questo è certamente il caso della generazione d'impulsi di breve durata (approssimativamente 8 μs), dove il software richiesto, come indicato in Fig. 4-2, è semplice. Tale software è, ad ogni modo, un po' più complesso di quello richiesto per generare impulsi di selezione

Fig. 4-2. Flowchart di un programma parziale necessario per generare un impulso di breve durata su ognuno dei bit della porta C.

dispositivo o indirizzi di selezione. Notare che nel flowchart riportato in Fig. 4-2, si è usata un'istruzione DCR A. Tale istruzione cambia la funzione della parola di controllo da set a reset.

La durata dell'impulso, nel caso sopra, sarà approssimativamente di 7,5 μs, per un microcomputer funzionante ad una frequenza di clock di 2 MHz. Tale è il tempo necessario per eseguire le istruzioni di decremento ed uscita. Impulsi di gate, di durata controllata, possono essere generati mediante l'inserimento di tempi di ritardo noti, consecutivi alla prima istruzione d'uscita. All'occorrenza, ad ogni modo, può essere più efficace posizionare un flip-flop esterno, impiegando due bit della porta C. L'impiego dei bit della porta C per operazioni di gate e reset hardware, è illustrato nell'Esperimento 4-2.

* Vedere Esperimenti con TTL e 8080 A volume 2: *Elettronica Digitale, Tecniche di Programmazione e Interfacciamento dei Microcomputer,* Capitolo 17.

4-2. PROCEDURA PER IL SET E RESET DEI BIT DELLA PORTA C

Per illustrare l'uso del byte di controllo di set/reset bit, scriviamo un programma per posizionare il bit PC5 della porta C. Dalla Fig. 4-1, i byte di controllo set/reset bit del bit PC5, sono costruiti come segue:

D7: Codice funzione = 0 per operazioni set/reset bit
D6,D5,D4: Non impiegati
D3,D2,D1: Selezione bit porta C = 101_2 per PC5
D0: Flag selezione set/reset = 1 per set bit
= 0 per reset bit

Segue che il formato della parola di controllo per il set/reset bit di PC5 è:

0 0 0 0 1 0 1 1/0
= 013_8 per set bit
= 012_8 per reset bit

Il diagramma di flusso per questo esempio è lo stesso di quello riportato in Fig. 4-2. Un listing del programma che è l'equivalente di questo flowchart, è mostrato nel programma 4-1. L'unica aggiunta riguarda l'inizializzazione del PPI, che deve essere effettuata nel programma, prima di un qualsiasi utilizzo del PPI.

Programma 4-1 (Fig. 4-3)

In questo programma non si è specificato il valore del byte di controllo del modo. A questo punto dovrebbe essere chiaro che il suo valore dipende dal modo in cui il PPI deve essere configurato. L'unica cosa che può essere detta per il momento, è che, poichè il bit PC5 è impiegato come uscita, il bit D3 (selezione I/O porta C superiore) della parola di controllo, è necessario che sia al valore logico 0 (Fig. 2-2A). Anche il codice del dispositivo del registro di controllo si è lasciato indefinito, dal momento che è funzione dell'hardware di decodifica d'indirizzo del PPI.

Altri tre punti, di questo programma di esempio sul funzionamento del set/reset bit, sono interessanti:

1. Alla locazione di memoria 003 010, si usa un'istruzione di un solo byte, DCR A, per cambiare il byte di controllo di set-bit, nell'accumulatore, con un byte di controllo di reset-bit. Ciò è possibile dal

```
                  /
                  /PROGRAMMA PER IL SET E RESET DEL BIT PC5 DELLA PORTA C
                  /
                  DB CNTRL 203
                  *003 00
003 000 078             MVIA     /CARICARE IN A
003 001                 MODE     /IL BYTE DI CONTROLLO DEL MODO
003 002 323             OUT      /INVIARLO IN USCITA
003 003 203             CNTRL    /CODICE DISPOSITIVO DEL REGISTRO DI
                                 /CONTROLLO
003 004 076       LOOP, MVIA     /CARICARE IN A IL BYTE DI CONTROLLO
003 005 013             013      /DI SET BIT PER PC5
003 006 323             OUT      /INVIARLO IN USCITA
003 007 203             CNTRL    /CODICE DISPOSITIVO DEL REGISTRO DI
                                 /CONTROLLO
003 010 075             DCRA     /DECREMENTARE A PER GENERARE UN BYTE

003 011 323             OUT      /INVIARE IL BYTE DI CONTROLLO
003 012 203             CNTRL    /DI RESET BIT PER PC5 AL REGISTRO DI
                                 CONTROLLO
003 013 303             JMP      /LOOP PER GENERARE UN ALTRO
                                 /IMPULSO
003 014 004             LOOP
003 015 003             0
```

Fig. 4-3. Programma 4-1.

momento che il byte di controllo di *reset*-bit è inferiore di uno a quello del *set*-bit.

L'impiego di questa istruzione ad un byte, al posto della seguente a due byte:

MVI A
BITRST

è consigliato, dal momento che permette risparmio di memoria che, all'occorrenza, può essere necessaria ed inoltre essa è lievemente più veloce (cinque cicli di clock in confronto ai sette per le istruzioni MVI A, BITRST).
L'istruzione di decremento non potrà certamente essere eseguita, se ci sono istruzioni che alterano il contenuto dell'accumulatore dopo l'istruzione d'uscita di set-bit e prima dell'istruzione d'uscita di reset-bit.

2. Gli impulsi generati dal programma 4-1, sono *attivi-bassi*, cioè lo stato logico di PC5, è normalmente basso e va alto momentaneamente.
Un impulso complementare, attivo basso, o invertito, può essere generato essenzialmente con lo stesso software, inviando in uscita prima un byte di controllo di reset-bit, *incrementando* l'accumulatore, poi inviando in uscita il risultante byte di controllo di set-bit. Facendo ciò, si è supposto che PC5 fosse inizialmente al valore logico uno.
Ora, l'Esperimento 4-1, mostrerà come avviene il caricamento del

registro di controllo del PPI con un byte di controllo del modo di *reset di tutti i bit della porta C al valore logico 0.* In questo programma, quindi, lo stato logico iniziale di PC5 è 0 e quindi, esso dovrà essere inizializzato al valore logico 1 impiegando un byte di controllo di set-bit, prima di entrare nel loop che inizia alla locazione 003 004. Ciò è illustrato nel programma 4-2. Le semplici variazioni software illustrate nel programma 4-2, evitano la necessità di un inverter ed inoltre, servono per illustrare il vantaggio di logica programmata (in questo caso il PPI ed il microcomputer), sostituendo l'hardware con il software.

3. Per generare un segnale di gate di durata più lunga, impiegando il programma 4-1, è necessaria una call ad una subroutine di tempo di ritardo, da inserirsi fra i due comandi d'uscita (Programma 4-2). Tale routine, dovrà poi ritardare il reset di PC5 per il tempo richiesto. Al ritorno dalla routine di ritardo, l'accumulatore dovrà essere caricato con la parola di controllo di reset-bit, per l'uscita verso il registro di controllo del PPI. L'impiego di una istruzione DCR A o MVI A, per fare ciò, dipende dal fatto che, la routine

Programma 4-2 (Fig. 4-4)

```
                    /
                    /QUESTO PROGRAMMA GENERA, SUL BIT PC5 DELLA PORTA C,
                    /UN IMPULSO ATTIVO BASSO LA CUI DURATA È CONTROLLATA
                    /CON UNA CHIAMATA ALLA SUBROUTINE DELAY
                    /
                    DB MODE 200  /QUESTA È UNA CONFIGURAZIONE ARBITRARIA
                    DB  CNTRL  203  /ANCHE  QUESTA  È  UNA  CONFIGURAZIONE
                                                            ARBITARIA
                    DW DELAY1 003 200  /INDIRIZZO D'INIZIO DELLA
                    DW DELAY2 003 220  /SUBROUTINE DELAY
                    *003 000
003 000 076        START, MVIA      /INIZIALIZZAZIONE DEL PPI
003 001 200               MODE
003 002 323               OUT
003 003 203               CNTRL
003 004 076               MVIA      /CARICARE IN A UN BYTE DI CONTROLLO
003 005 013               013       /PER PORRE PC5 AL VALORE LOGICO UNO
003 006 323               OUT
003 007 203               CNTRL
003 010 076        LOOP,  MVIA      /CARICARE IN A UN BYTE DI CONTROLLO
                                    /PER PORRE PC5
003 011 012               012       /AL VALORE LOGICO ZERO
003 012 323               OUT
003 013 203               CNTRL
003 014 315               CALL      /CHIAMATA ALLA SUBROUTINE DELAY1 PER
                                    /GENERARE
003 015 200               DELAY1    /L'AMPIEZZA D'IMPULSO NECESSARIA
003 016 003               0
003 017 074               INRA      /INCREMENTARE A PER OTTENERE
                                    /IL BYTE DI CONTROLLO
                                    /PER IL SET DI PC5
003 020 323               OUT
003 021 203               CNTRL
```

```
003 022 315            CALL      /CHIAMATA ALLA SUBROUTINE DELAY2
                                 /PER GENERARE
003 023 220            DELAY2    /IL PERIODO FRA GLI IMPULSI
003 024 003            0
003 025 303            JMP
003 026 010            LOOP
003 027 003            0
```

Fig. 4-4. Programma 4-2.

tempo di ritardo può distruggere o meno il contenuto dell'accumulatore. Ovviamente, l'istruzione MVI A, dovrà essere usata se il contenuto dell'accumulatore sarà stato distrutto. Ad ogni modo, è pratica comune, scrivendo subroutine, quella di salvare lo stato di tutti i registri (impiegando istruzioni di PUSH) che vengono usati dalla subroutine e di ripristinare il loro stato iniziale (impiegando istruzioni di POP), prima del ritorno della subroutine. Sarà quindi in genere possibile, l'impiego di un'istruzione DCR A. Controllare la subroutine prima di prendere la decisione.

4-3 ESEMPIO: CONTROLLO DI UNA VALVOLA

(A) Il processo fisico e la sua interfaccia di controllo

In questo paragrafo, è riportato un esempio pratico del modo in cui può essere usata la possibilità di set/reset bit della porta C, per controllare e mettere in sequenza le operazioni di un'interfaccia a microcomputer. L'esempio riguarda il controllo delle condizioni di vuoto in un reattore chimico. La pressione assoluta nel reattore, deve essere ridotta approssimativamente a circa 5×10^{-3} mm di mercurio (cioè, 5×10^{-3} torr), valore questo, che può essere convenientemente raggiunto impiegando una pompa per il vuoto. La pressione nel recipiente del reattore, è misurata impiegando un termistore di misura della pressione, che è posto nel condotto per il vuoto del recipiente.

La Fig. 4-5, mostra uno schema della linea per il vuoto e l'interfaccia di controllo al PPI. Fra la pompa ed il reattore è stata inserita una valvola a solenoide, in modo che il reattore sia isolato dalla pompa, una volta che si sia raggiunta la condizione di vuoto. La sequenza di eventi controllati dal microcomputer, è la seguente:

- Accensione della pompa per il vuoto.
- Apertura della valvola di pressione a solenoide.
- Visualizzazione della pressione.

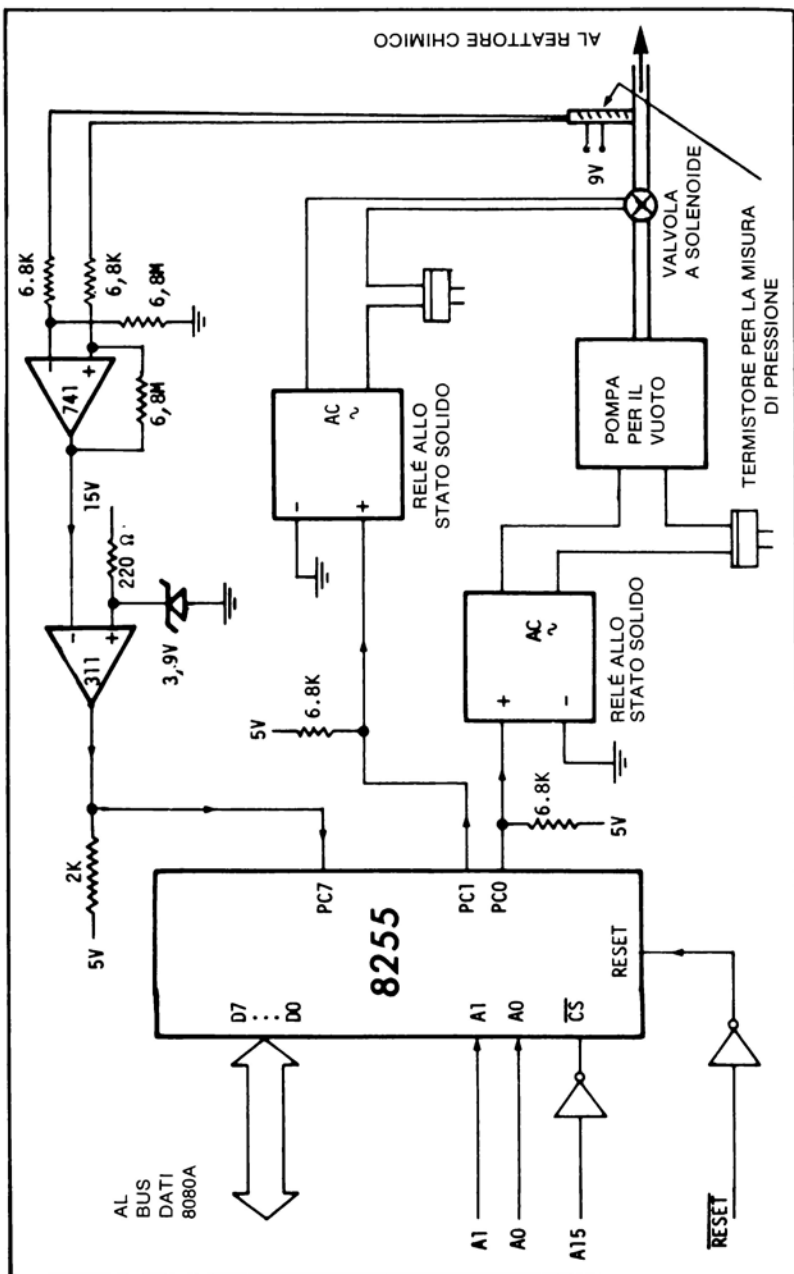



Fig. 4-5. Interfaccia di controllo del vuoto.

- Chiusura della valvola a solenoide, quando la pressione raggiunge il valore richiesto (in questo caso 5×10^{-3} torr).

Sia la pompa che la valvola a solenoide, sono alimentate a 110 V_{ca} e possono quindi, essere accese e spente mediante relè allo stato solido. Tali dispositivi sono economici (pochi dollari) e possono mantenere 10 Ampere a 110 Volt, con una tensione cc di commutazione fra 3 e 32 V sui loro ingressi cc. La condizione normale della valvola di pressione è chiusa. Quando si applicano i 110 V_{ca}, una bobina di un solenoide riceve energia, e un listello di metallo, che è connesso ad una molla caricata della valvola, viene fatto salire per aprire la valvola. Quest'ultima, resta aperta per tutto il tempo in cui la tensione è applicata al solenoide. Da tutto ciò, segue che, nell'interfaccia illustrata in Fig. 4-5, i bit PC0 e PC1 della porta C, vengono impiegati per pilotare due relè allo stato solido, che sono collegati rispettivamente alla pompa ed al solenoide della valvola.

Un'interessante caratteristica delle porte B e C del PPI, che viene sfruttata qui, è che *ogni gruppo di otto linee può fornire circa 2 mA di corrente di pilotaggio, quando la tensione d'uscita è 1,5 V.* Tale corrente di pilotaggio, anche quando le uscite sono più grandi di 3 V (impiegando la resistenza di pull-up), è sufficiente per pilotare gli ingressi cc dei relè allo stato solido e di accenderli quando le uscite PC0 e PC1, sono al valore logico alto. Ciò evita la necessità di buffer fra le uscite del PPI ed i relè allo stato solido.

Il termistore per la misura della pressione, più propriamente trasduttore, ha una tensione d'uscita che non è adatta per un collegamento diretto al PPI (in questo caso PC7). Alla pressione atmosferica la sua tensione d'uscita è di circa 120 mV, mentre ad una pressione assoluta di circa 5×10^{-3} torr, la sua tensione d'uscita è di circa 4 mV. Tali potenziali vengono detti "floating" (fluttuanti), poichè non sono riferiti a una massa (0 V). Gli amplificatori operazionali di Fig. 4-5, sono usati per produrre un segnale d'ingresso al PC7 del PPI, il quale è normalmente a 0 V (valore logico 0) e sale a +5 V (valore logico 1), quando la pressione del sistema raggiunge 5×10^{-3} torr.

(B) Il PPI e il programma di controllo

In questo esempio il PPI è stato collegato ad un microcomputer basato sull'8080A, per I/O in mappa di memoria. In base alla discussione fatta

nel paragrafo 2-1, si può vedere che gli *indirizzi dei dispositivi*, in questo caso, sono:

PORTA C: 200 002
REGISTRO DI CONTROLLO: 200 003

Le label PORT C e CNTRL, nel programma di controllo il cui listing è riportata in Fig. 4-6, sono state definite con quei valori.

Programma 4-3 (Fig. 4-6)

Dobbiamo ora costruire il *byte di controllo del modo*, MODE, che configurerà la porta C superiore come ingresso per accettare il flag di pressione; tale parola configura anche la porta C inferiore come uscita, cosicchè i due relè allo stato solido, possano essere accesi e spenti. Sebbene non vengano impiegate, le porte A e B saranno configurate come ingressi in modo 0. Il byte di controllo del modo necessario, può essere costruito dalla Fig. 2-2A ed è:

```
                  /
                  /PROGRAMMA DI CONTROLLO DELLA VALVOLA PER IL VUOTO
                  /
                  DB MODE 232
                  DW PORTC 200 002
                  DW CNTRL 200 003
                  DW DELAY 003 200
                  *003 000
003 000 061      START, LWISP    /IMPOSTARE LO STACK POINTER
003 001 000             000
003 002 004             004
003 003 041             LXIH     /IMPOSTARE L'INDIRIZZO DISPOSITIVO
003 004 003             CNTRL    /DEL REGISTRO DI CONTROLLO DEL PPI
003 005 200             0
003 006 021             LXID     /CARICARE NELLA COPPIA DI REGISTRI D
003 007 003             003      /BYTE DI CONTROLLO SET BIT PER PC1 IN D
003 010 001             001      /BYTE DI CONTROLLO SET BIT PER PC0 IN E
003 011 076             MVIA     /CARICARE IN A IL BYTE
003 012 232             MODE     /DI CONTROLLO DEL MODO
003 013 167             MOVMA    /INVIARLO AL REGISTRO DI CONTROLLO
003 014 163             MOVME    /AVVIARE LA POMPA
003 015 162             MOVMD    /APRIRE LA VALVOLA A SOLENOIDE
003 016 053             DCXH     /IMPOSTARE L'INDIRIZZO DELLA PORTA C
                                 /IN H,L
003 017 176      LOOP,  MOVAM    /INSERIRE IL FLAG DI PRESSIONE
003 020 346             ANI      /MASCHERARE PC7
003 021 200             200
003 022 312             JZ       /SI È RAGGIUNTO IL VUOTO
003 023 017             LOOP     /NO, CONTROLLARE DI NUOVO
003 024 003             0
003 025 315             CALL     /SI, ATTENDERE UN SECONDO
003 026 200             DELAY
003 027 003             0
003 030 176             MOVAM    /E CONTROLLARE NUOVAMENTE
003 031 346             ANI      /MASCHERARE PC7
003 032 200             200
003 033 312             JZ       /SI È ANCORA STABILITO IL VUOTO
003 034 017             LOOP     /NO, CONTROLLARE DI NUOVO
003 035 003             0
```

```
003 036 043          INXH      /SI, IMPOSTARE L'INDIRIZZO DEL
                               /REGISTRO DI CONTROLLO
003 037 172          MOVAD     /COSTRUIRE IL BYTE DI CONTROLLO DEL
                               /RESET BIT PC1
003 040 075          DCRA
003 041 167          MOVMA     /ED USARLO PER CHIUDERE LA VALVOLA
               /
               /
               /
               /
               /
```

Fig. 4-6. Programma 4-3.

MODE 232

Poichè i bit PC0 e PC1 della porta C devono essere posti a 0 e ad 1, per controllare rispettivamente la pompa ed il solenoide della valvola, saranno necessari *byte di controllo set/reset bit*. Definiremo i byte di controllo di set bit per PC0 (PC0SET) e PC1 (PC1SET) e decrementeremo i loro rispettivi valori nel programma, per ottenere i byte di controllo di reset bit. Dalla Fig. 2-2B, si ha che i valori richiesti sono:

PC0SET 001

PC1SET 003

Ricordando ora che il PPI è collegato al microcomputer per mezzo di I/O in mappa di memoria, si riporta un listing, Programma 4-3 (Fig. 4-6), di un programma che stabilisce le condizioni di vuoto nel reattore, in base alla sequenza di eventi sopra descritti. Dopo l'inizializzazione del PPI, con la porta C superiore configurata come ingresso e la porta C inferiore come uscita, il programma accende la pompa e apre la valvola a solenoide, ponendo rispettivamente al valore logico 1 PC0 e PC1. Il flag di pressione viene quindi visualizzato introducendo il contenuto della porta C e mascherando tutti i bit eccetto PC7. Quando PC7 è al valore logico 1, si genera un ritardo di 1 secondo ed il flag è nuovamente controllato come descritto sopra. Poichè la pressione del sistema cambia molto lentamente avvicinandosi a 5×10^{-3} torr, il rilevamento iniziale del valore logico 1 di PC7, può rappresentare solamente un'escursione di transitorio della pressione del sistema, verso 5×10^{-3} torr. Il ritardo assicura che si sia fissato un valore stabile della pressione richiesta. Nel caso suddetto, la valvola viene chiusa resettando, al valore 0, il bit PC0 della porta C.

La caratteristica di programmazione, in questo caso, è l'impiego dell'istruzione di riferimento alla memoria MOV M, r, per inviare fuori i byte di set-bit e reset-bit al PPI, il quale è collegato come una porta I/O in mappa di memoria. In questo caso si è usata la coppia di registri H, per

memorizzare inizialmente l'indirizzo del dispositivo del registro di controllo del PPI. L'indirizzo della porta C, è poi ottenuto decrementando la coppia di registri H con l'istruzione DCX H. Alla fine del programma, la coppia di registri H è incrementata (INX H), per stabilire di nuovo l'indirizzo del dispositivo del registro di controllo, cosicchè si possa inviare, al registro di controllo del PPI, un byte di controllo di reset-bit per PC1, per diseccitare la valvola. Bisogna porre cura nello stabilire l'indirizzo del dispositivo per il PPI tramite tale tecnica di incremento e decremento, poichè il rischio, che esiste sempre, è quello di perdere traccia di "dove si stia andando", o che il contenuto della coppia di registri possa essere alterato in seguito a una call ad una subroutine. Notare anche che i registri D ed E, sono usati per memorizzare rispettivamente i byte di controllo set-bit di PC1 e PC0. Tali byte vengono poi inviati direttamente al registro di controllo del PPI con le istruzioni MOV M, D e MOV M, E.

In conclusione, l'esempio ha illustrato l'impiego della caratteristica di set/reset bit della porta C, per il controllo e la messa in sequenza delle operazioni di una linea per il vuoto. L'hardware è ridotto al minimo tramite la possibilità di porre ad 1, e più tardi porre a 0, i bit della porta C. L'alta corrente di pilotaggio di ogni bit della porta C, al valore di 1 logico in uscita, elimina la necessità di buffer di pilotaggio dove i dispositivi debbono essere pilotati al valore logico 1. (Notare che il fan-out di ciascuno dei bit della porta C, al valore logico 0, è di un solo TTL standard di carico). Il software è lineare e comprende l'inizializzazione del PPI, l'invio di byte di controllo di set-bit e reset-bit al registro di controllo del PPI ed il polling della porta C per conoscere lo stato logico di PC7.

4-4. SOMMARIO DEGLI ESPERIMENTI 4-1 E 4-2

Esperimento	*Descrizione*
4-1	Questo esperimento, mostra come può essere usata la *parola di controllo set/reset bit*, per posizionare individualmente i bit della porta C; mostra anche come la *parola di controllo codice* azzera tutti i bit della porta C.
4-2	In questo esperimento: (a) si impiegheranno le operazioni di set/reset bit del PPI, per inserire impulsi in un contatore. (b) Si preleveranno dati dal contatore impiegando I/O semplice in modo 0 (c) Si azzererà il contatore impiegando la possibilità di set/reset bit della porta C.

ESPERIMENTO 4-1
SET E RESET DEI BIT DELLA PORTA C

Scopo

Gli scopi di questo esperimento sono i seguenti:

(a) Posizionare individualmente i bit della porta C impiegando il formato set/reset della parola di controllo.

(b) Mostrare che i bit della porta vengono azzerati da:
   - (i) Un valore logico alto al pin di reset del PPI e
   - (ii) Inviando un byte di controllo del modo al registro di controllo.

Passo 1

Collegare il circuito mostrato nello schema di Fig. 4-7 e caricare il programma nella memoria di lettura/scrittura del microcomputer.

Passo 2

Inserire un'istruzione di halt (HLT = 166), per la prima istruzione di NOP all'indirizzo 003 011. Avviare il programma. Si è accesa qualcuna delle lampade alla porta C?

Sì, noi abbiamo visto PC0 alto (valore logico 1). Ciò è coerente con lo schema indicato in Fig. 4-1?

Si, infatti un byte di controllo set/reset bit di valore 001 pone ad 1 il bit PC0.

Passo 3

Siete in grado di determinare i byte di controllo set/reset bit, necessari per porre, uno alla volta, i bit PC3, PC5 e PC6, al valore logico 1? (vedere Fig. 4-1).

Schema del circuito (Fig. 4-7)

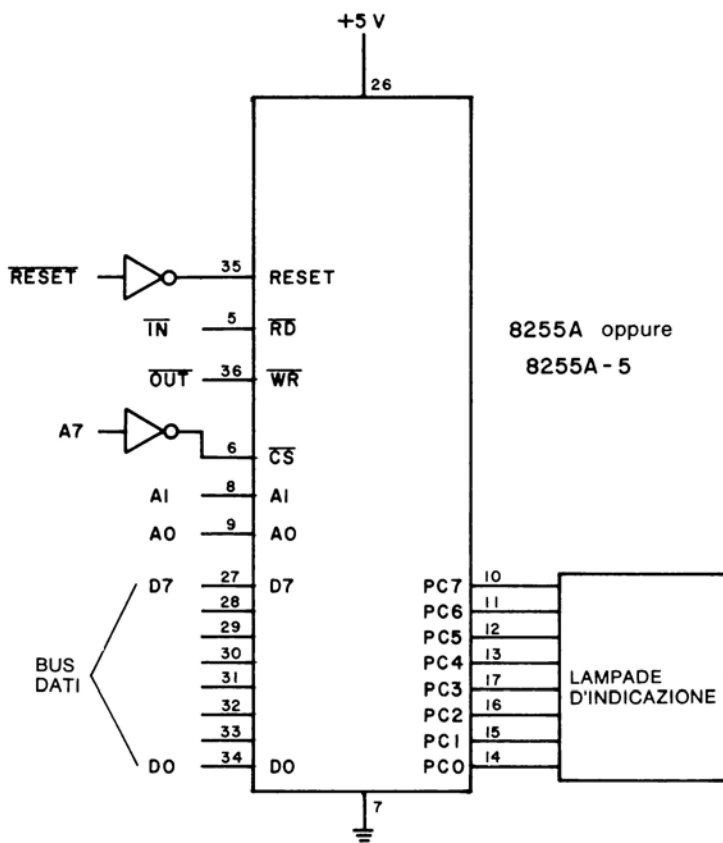



Fig. 4-7. Circuito per l'Esperimento 4-1.

Programma (Fig. 4-8)

```
                  /
                  /QUESTO PROGRAMMA ILLUSTRA IL SET ED
                  /IL RESET DEI BIT DA PC0 A PC7 DELLA PORTA C
                  /
                  DB CNTRL 203
                  *003 000
003 000 016             MVIC       /CARICARE NEL REGISTRO C IL
003 001 200             200        /BYTE DI CONTROLLO DEL MODO
003 002 171             MOVAC      /COPIARE IL BYTE DI CONTROLLO DEL
                                   /MODO IN A
```

```
003 003 323        OUT      /INVIARLO AL REGISTRO DI
003 004 203        CNTRL    /CONTROLLO DEL PPI
003 005 076        MVIA     /CARICARE IN ACCUMULATORE LA
                            /PARÓLA
003 006 001        001      /DI CONTROLLO SET BIT
003 007 323        OUT      /INVIARLA AL REGISTRO DI
003 010 203        CNTRL    /CONTROLLO DEL PPI
003 011 000        NOP
003 012 000        NOP
003 013 000        NOP
003 014 075        DCRA     /DECREMENTARE A
003 015 323        OUT      /INVIARE IL BYTE DI CONTROLLO RESET
                            /BIT
003 016 203        CNTRL    /AL REGISTRO DI CONTROLLO
003 017 166        HLT
```

Fig. 4-8. Programma per l'Esperimento 4-1.

Byte di controllo del modo

PC3

PC4

PC5

Noi abbiamo usato, rispettivamente, i byte di controllo 007, 013 e 015. Inserire tali valori del byte di controllo set/reset bit, uno alla volta, all'indirizzo 003 006 e fare eseguire il programma. I corrispondenti bit vengono posti al valore logico 1?

Sì, ma solamente uno alla volta. I bit PC3, PC5 e PC6, non vengono posti simultaneamente al valore 1 logico. Come deve essere fatto un programma che, una volta eseguito, lasci tutti tre i bit al valore logico 1?

Si suggerisce la sequenza indicata in Fig. 4-9. Tale sequenza lascerà alti tutti i bit, non appena il programma verrà eseguito.

```
MVIA     /PORRE PC3 AD 1
007
OUT
203
MVIA     /PORRE PC5 AD 1
013
OUT
203
MVIA     /PORRE PC6 AD 1
015
OUT
203
HLT
```

Fig. 4-9. Sequenza per porre PC3, PC5 e PC6 al valore logico 1.

Passo 4

Togliere l'istruzione di halt all'indirizzo 003 011, sostituendola, insieme alle due istruzioni NOP che la seguono, con una call alla subroutine DELAY:

```
003 011   315   CALL
003 012   200   DELAY
003 013   003     ø
```

Inserire la subroutine di ritardo, DELAY, il cui listing è riportato in Fig. 3-10, se non è già stata introdotta nel microcomputer.

Passo 5

Assicurarsi di reinserire il byte di controllo set bit PC0, all'indirizzo 003 006. Avviare il microcomputer ed osservare ogni cambiamento sul monitor. Cosa accade?

La lampada rivelatrice del bit PC0, si accende (valore logico 1), poi si spegne. Il periodo di accensione è di circa 0,1 secondi. Perchè si osserva tale fenomeno?

Si invia un byte di controllo set bit al PPI, poi si esegue la subroutine del tempo di ritardo. Dopo che il tempo di ritardo è terminato, il programma invia in uscita, al PPI, un byte di controllo reset bit. Il tempo di accensione può essere variato cambiando i byte di temporizzazione nella subroutine DELAY.

Passo 6

Per mostrare il controllo software del tempo di accensione, cambiare i byte di temporizzazione nella subroutine DELAY con 377 377. Provare di nuovo il programma. Che cosa si vede?

Il tempo di accensione dovrebbe essere di circa 2 secondi.

Passo 7

In questo passo si esaminerà l'effetto di un reset esterno, sullo stato logico delle uscite della porta C. Per vedere tale effetto, sostituire l'istru-

zione CALL (315), all'indirizzo 003 011, con un'istruzione halt (HLT). Caricare il byte di controllo set bit, 001, alla locazione 003 006. Ciò porrà il bit PC0 al valore logico 1. Avviare il programma. Cosa succede? Eseguire un reset del computer con il suo pulsante o interruttore di reset. Che cosa succede ora? Perchè?

Mentre il programma veniva eseguito, abbiamo visto la lampada indicatrice di PC0 accendersi (1 logico). Quando viene eseguito il reset essa si spense. Tale caratteristica della funzione di reset del PPI, è comune a tutti i bit della porta C. Si può avere conferma di ciò, ripetendo tale passo con diversi byte di controllo di set bit, posti alla locazione 003 006.

Passo 8

Un altro metodo per il reset di tutti i bit della porta C al valore logico 0, è di inviare un byte di controllo del modo al registro di controllo del PPI. Questa non è certamente la principale caratteristica del byte di controllo del modo, ma è un utile ed interessante aspetto. Per illustrare quanto detto, eseguire le seguenti variazioni nel programma:

(a) Reinserire l'istruzione CALL (315) alla locazione 003 011.
(b) Impiegare, nella subroutine del tempo di ritardo, i byte 377 377.
(c) Cambiare gli step del programma dalla locazione 003 014 in avanti, come in Fig. 4-10.

Eseguire un reset del microcomputer ed avviare il programma. Che cosa si osserva?

```
003 014 076        MVIA     /CARICARE IN ACCUMULATORE IL
003 015 200        200      /BYTE DI CONTROLLO DEL MODO
003 016 323        OUT      /E INVIARLO
003 017 203        CNTRL    /AL REGISTRO DI CONTROLLO
003 020 166        HLT
```

Fig. 4-10. Passi del programma alterato.

Noi abbiamo visto che la lampada indicatrice di PC0 si è accesa, poi subito dopo, si è spenta. Come si può spiegare ciò, dal momento che nel programma non c'era alcun byte di controllo reset bit?

Ogni variazione del byte di controllo inviata all'8255, in cui il bit D7 è al valore 1 logico, pone tutti i bit della porta C al valore logico 0. Siete in grado di indicare perchè tale caratteristica è utile?

Quando si carica un byte di controllo del modo ne! registro di controllo del PPI, tutti i bit della porta C sono *posti al valore logico 0.* Tale fatto è utile poichè definisce lo stato logico iniziale di tutti i bit della porta C, dopo che il PPI è stato inizializzato con un byte di controllo del modo. Se una linea di controllo d'interfaccia che si sta pilotando, deve essere inizializzata al valore logico 0, in seguito a tale caratteristica, non richiede nessun'altra azione consecutiva all'inizializzazione del PPI. Se, d'altra parte, lo stato logico iniziale della linea di controllo d'interfaccia deve essere un 1 logico, il bit della porta C che guida la linea di controllo, deve essere posto al valore logico 1 con un byte di controllo di set bit successivo all'inizializzazione del PPI.

Domande

1. Usando il formato della parola di controllo del modo riportato in Fig. 2-2A, descrivere il modo in cui le porte del PPI, da A fino a C, sono state configurate dalla parola di controllo del modo usata in questo esperimento. Nell'esperimento, la porta C può essere stata configurata come ingresso? Se no, perchè?

2. Spiegare perchè, in questo esperimento, si è usato un codice dispositivo del registro di controllo di valore 203.

3. Quale vantaggio è derivato dall'invio in uscita di un byte di controllo del modo, all'inizio di un programma, per il reset dei bit della porta C?

4. Indicare alcuni usi della caratteristica di set/reset bit dell'8255.

ESPERIMENTO 4-2
UN DATA LOGGER BASATO SUL PPI

Scopo

Lo scopo di questo esperimento è di illustrare come la caratteristica del PPI, di set/reset bit, possa essere usata per generare impulsi per circuiti di gate e per operazioni di reset. Si costruirà un semplice *data logger* basato sul PPI in modo 0.

Passo 1

Collegare il circuito mostrato nello schema di Fig. 4-12. Notare che, poichè era necessario un gate NAND per inserire gli impulsi di clock attraverso l'ingresso A_{IN} del primo SN7493, i rimanenti gate del chip SN7400 sono stati usati per fornire gli inverter necessari al circuito.

Passo 2

Inserire il programma A nella memoria di lettura/scrittura del microcomputer. Porre la frequenza del clock esterno (collegato ai contatori SN7493) a un valore da 1 a 5Hz.

Programmi (Fig. 4-11)

```
                /
                /PROGRAMMA "B"
                /DESCRIZIONE: QUESTO È UN PROGRAMMA
                /             PER UN DATA LOGGER
                /             BASATO SUL PPI
                /
                DB MODE 202
                DB CNTRL 203
                DW DELAY 003 200
                *003 000
003 000 061         LXISP
003 001 000         000
003 002 004         004
003 003 076         MVIA       /INIZIALIZZARE IL PPI IN MODO 0 CON
003 004 202         MODE       /LE PORTE A E C = USCITE
                               /PORTA B = INGRESSO
003 005 323         OUT
003 006 203         CNTRL
003 007 076         MVIA       /PORRE PC0 COME ABILITAZIONE
                               /COUNTER
003 010 001         001
003 011 323         OUT
003 012 203         CNTRL
003 013 076         MVIA       /APRIRE IL GATE MEDIANTE IL SET DI PC7
003 014 017         017
003 015 323         OUT
003 016 203         CNTRL
003 017 315         CALL       /ATTENDERE UN
003 020 200         DELAY      /CONTEGGIO
003 021 003         0
```

```
003 022 076             MVIA      /CHIUDERE IL GATE MEDIANTE IL RESET DI
                                  /PC7
003 023 016             016
003 024 323             OUT
003 025 203             CNTRL
003 026 333             IN        /INSERIRE I DATI DALLA
003 027 201             201       /PORTA B
003 030 323             OUT       /INVIARE I DATI INVARIATI ALLA
003 031 200             200       /PORTA A
003 032 166             HLT

                 /
                 /PROGRAMMA "A"
                 /
                 /DESCRIZIONE: QUESTO PROGRAMMA È USATO PER
                 /CONTROLLARE
                 /IL FUNZIONAMENTO SODDISFACENTE DEL CIRCUITO
                 /D'INTERFACCIA
                 /
                 DB MODE 202
                 DB CNTRL 203
                 *003 000
003 000 076             MVIA      /IMPOSTARE IL MODO DEL PPI PER
003 001 202             MODE      /A = USCITA, B = INGRESSO
003 002 323             OUT       /C̄ = USCITA
003 003 203             CNTRL
003 004 076             MVIA      /IMPOSTARE PC0 AL VALORE LOGICO 1 PER
003 005 001             001       /ABILITARE I COUNTER
003 006 323             OUT
003 007 203             CNTRL
003 010 076             MVIA      /APRIRE IL GATE MEDIANTE
003 011 017             017       /IL SET DI PC7
003 012 323             OUT
003 013 203             CNTRL
003 014 333      LOOP,  IN        /INSERIRE I CONTEGGI
003 015 201             201       /DALLA PORTA B
003 016 323             OUT       /INVIARLI IN USCITA
003 017 200             200       /ALLA PORTA A
003 020 303             JMP
003 021 014             LOOP
003 022 003             0
```

Fig. 4-11. Programma per l'Esperimento 4-2.

Configurazione del pin del chip circuiti integrati (Fig. 4-12)

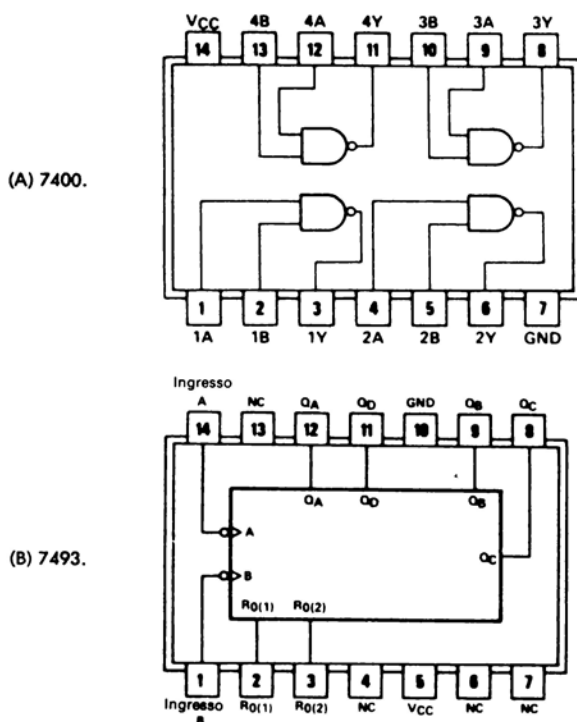



Fig. 4-12. Configurazione del pin del chip IC.

Passo 3

Avviare il programma. Si nota qualche variazione sui LED collegati alla porta A?

Noi vediamo un conteggio che avviene alla stessa frequenza del clock impostato nel passo 2. Questo rappresenta un controllo dell'interfaccia e del programma. Se non si osserva quanto sopra, tornare indietro e controllare programma e interfaccia. Se si varia la frequenza di clock, si deve vedere una variazione simile, sia nel conteggio alla porta A, sia nella frequenza a cui le lampade indicatrici vengono incrementate.

Schema del circuito (Fig. 4-13)

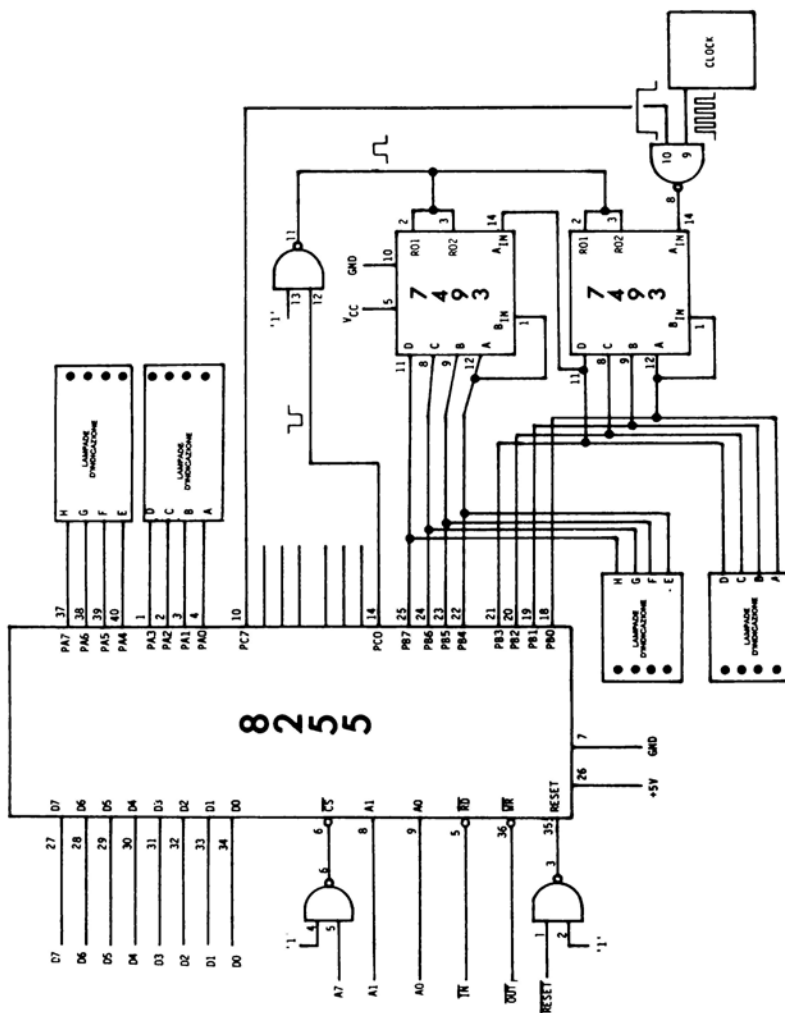



Fig. 4-13. Circuito per l'Esperimento 4-2.

Passo 4

Caricare il Programma B in memoria. Studiare tale programma ed i commenti forniti unitamente allo schema del circuito e disegnare, nello spazio qui di seguito, un diagramma di flusso del funzionamento del programma. Per aiutarvi, i commenti in questo caso sono più generalizzati. Non è necessario disegnare un flowchart della subroutine DELAY. NOTA: Se non si è completato l'Esperimento 4-1, sarà necessario inserire in memoria la subroutine di ritardo, DELAY, il cui listing è riportato in Fig. 3-10.

A questo punto, vi sarete accorti che lo scopo del programma è di registrare un conteggio, ad 8 bit, del numero di impulsi che passano attraverso il gate SN7400 mentre è aperto, in seguito visualizzare tale conteggio alla porta A. Il programma innanzi tutto, pone a 0 le linee di reset dell'SN7493, mediante il bit PC0 del PP1. Si porta poi, al valore logico 1, il bit PC7 del PP1 per aprire il gate e permettere quindi il conteggio degli impulsi. Il bit PC7 viene azzerato per fermare il conteggio, gli 8 bit dei due contatori SN7493, vengono inseriti e visualizzati alla porta A. I contatori vengono azzerati con PC0.

Passo 5

Impostare il clock ad una frequenza di circa 50-60 Hz. Se non si possiede un oscilloscopio per fare ciò, seguire i seguenti passi:

(a) Sconnettere una lampada di rivelazione dal PPI e ricollegare l'indicatore all'uscita del clock. Regolare la frequenza del clock in modo che scompaia l'effetto flicker (sfarfallamento) dell'indicatore.

(b) Ricollegare la lampada di rivelazione all'apposito pin del chip 8255.

Impostare nella maniera seguente, i byte di temporizzazione della subroutine DELAY:

```
003 203   060 { Tali byte generano un ritardo
003 204   165 { di 1 secondo
```

Avviare il programma. Che cosa si osserva sui due gruppi di lampade di rivelazione?

Noi abbiamo visto che sulla porta A e sulle lampade collegate ai contatori SN7493, era indicato il medesimo conteggio. Tale risultato era 00111000_2, ovvero 56_{10}. Questo valore, sarà probabilmente diverso dal vostro, dal momento che dipende dalla frequenza di clock. Che cosa rappresenta tale conteggio?

Il valore letto, rappresenta il numero d'impulsi che i contatori SN7493 hanno rilevato, durante l'intervallo di tempo trascorso fra "l'apertura" del gate (step 003 016) e la sua "chiusura" (step 003 025). la subroutine DELAY, tiene aperto il gate per circa 1 secondo.

Passo 6

Ripetere nuovamente il passo 5 eseguendo il programma e annotare, qui sotto, il risultato del conteggio. Tale valore è vicino a quello trovato nel passo 5?

Abbiamo di nuovo visto il valore 56_{10}, sia sulle lampade alla porta A, che su quelle collegate ai contatori SN7493. Se il gate è aperto per 1 secondo, quale operazione compie il microcomputer?

Esso esegue una *misura di frequenza*, poichè rileva e visualizza il numero d'impulsi ricevuti in un secondo; ciò è, per definizione, la frequenza del treno d'impulsi del clock esterno.

Passo 7

Sfortunatamente il programma B, esegue una sola misura. Si può modificare il programma per fare una misura continua che possa essere usata per aggiornare il display? Indicare variazioni semplici del programma che permettano ciò.

Abbiamo cambiato il programma in modo che, dopo aver inviato il valore accumulato alla porta A, i contatori siano azzerati ed il programma ricicli continuamente attraverso gli appositi step. L'istruzione di halt (HLT) alla locazione 003 032, è stata sostituita con le istruzioni di Fig. 4-14. Effettuare i cambiamenti necessari al proprio programma ed eseguire una prova. Quando si regola la frequenza di clock, il display indica qualche variazione? Dovrebbe farlo! Si ricordi comunque che il campo di questo misuratore di frequenza va da 0 a 255 Hz.

```
003 032 076        MVIA     /RESET DEL COUNTER MEDIANTE IL RESET
003 033 000        000      /DI PC0
003 034 323        OUT
003 035 203        CNTRL
003 036 303        JMP      /RICICLARE PER RIPETERE IL
003 037 004        LOOP     /CICLO DI DATA LOGGER
003 040 003        0
```

Fig. 4-14. Istruzioni da sostituire all'istruzione HALT.

Domande

1. Come si può cambiare l'intero sistema, per misurare frequenze più alte? (Per tale problema esiste una soluzione hardware ed una software).

2. Perchè le linee di reset dei contatori SN7493 sono pilotate per mezzo di inverter, anzichè direttamente da PC0?

CAPITOLO 5

HANDSHAKING DI I/O A COMANDO DI STATO: FUNZIONAMENTO COMBINATO IN MODO 0 E SET/RESET BIT

5-1 CHE COSA SIGNIFICA HANDSHAKING?

Nel Capitolo 3, si è fatta una classificazione delle operazioni di ingresso/uscita secondo due categorie, chiamate trasferimento incondizionato e condizionato di dati. Il trasferimento incondizionato di dati, è il più semplice, poichè presuppone che il dispositivo di I/O sia in grado di ricevere o fornire i dati immediatamente, ad ogni richiesta del microcomputer. Il trasferimento condizionato o asincrono di dati, che sarà trattato in questo capitolo, è così chiamato a causa della mancanza di una base di sincronizzazione del tempo, fra un microprocessore e alcune delle sue periferiche. Il trasferimento dati è quindi "condizionato", allo stato di prontezza della periferica. Tali problemi di temporizzazione sorgono, in genere, a causa del piccolo, ma finito, tempo richiesto da parte di molte periferiche, sia a fornire dati su richiesta che ad accettare e memorizzare i dati loro presentati. I convertitori analogici digitali (A/D) ed i lettori di nastri di carta, ad esempio, richiedono entrambi un ritardo successivo all'inizio di un comando di lettura, prima che il dato sia disponibile per l'ingresso. Per un convertitore A/D, è necessario del tempo per la conversione analogico digitale, mentre, per un lettore di nastro di carta, si richiede del tempo per il movimento meccanico del nastro. Esistono situazioni analoghe per i dispositivi d'uscita. Un perforatore di nastri di carta, ad esempio, necessita di tempo per muovere e perforare il nastro, allo stesso modo una stampante ha bisogno di tempo,

dopo la ricezione del dato, per stampare il carattere decodificato.

Tali problemi di temporizzazione di I/O, vengono superati in pratica, fornendo le periferiche di *linee di stato*, che possano essere usate per indicare al microcomputer lo stato attuale dei dispositivi di I/O. Queste linee di stato o flag, sono in genere sotto forma di flip-flop che indicano, fra le altre cose, se la periferica è *pronta* per un trasferimento dati, o se è *occupata* a completare un trasferimento dati iniziato precedentemente. Il microcomputer a sua volta, deve essere in grado di comunicare alle sue

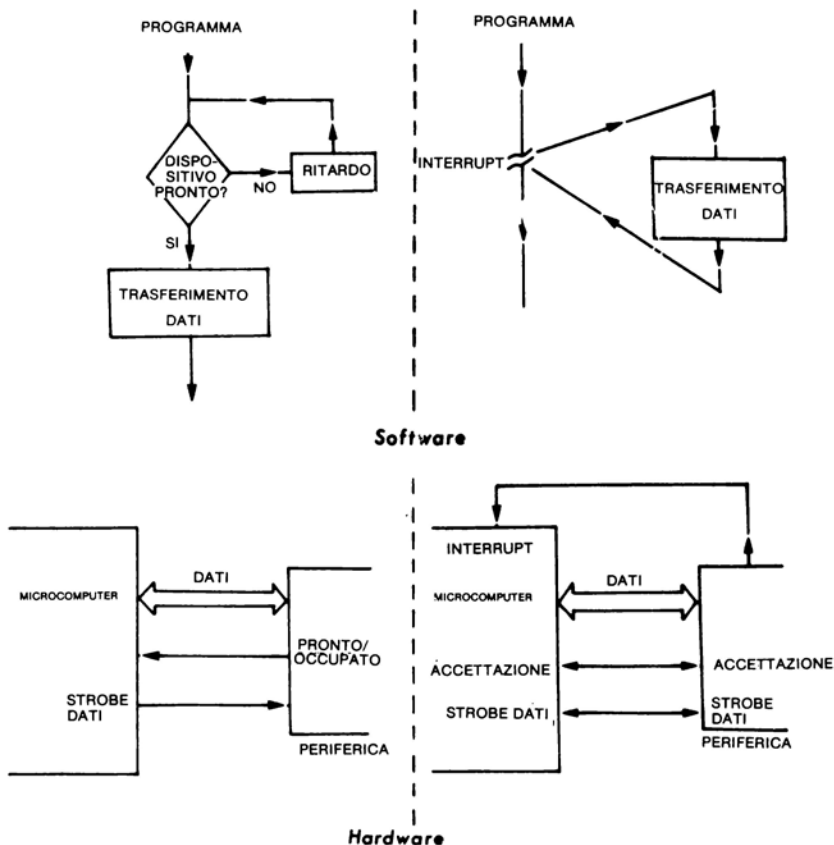



Fig. 5-1. Differenze essenziali fra handshaking di I/O a comando di stato e a comando di interrupt.

periferiche, quando vuole iniziare un trasferimento di dati. Esso esegue ciò per mezzo di un impulso di *strobe*.

Nel caso più semplice, quindi, ci sono due linee di controllo, in aggiunta alle linee dati, necessarie a collegare un microcomputer ed una periferica "lenta". Una linea, la *linea di strobe*, è fornita dal microcomputer, mentre la seconda, la linea di *occupato-pronto*, è fornita dalla periferica, come mostrato in Fig. 5-1A. Si assicura quindi, un ordinato flusso di dati fra microcomputer e periferica, mediante lo scambio di impulsi su queste due linee. Per leggere dati da una periferica, ad esempio, il microcomputer invia un impulso di "strobe dati", sulla sua linea di strobe, per iniziare il processo di conversione dati o raccolta di dati. La periferica risponde con un livello logico di "occupato" sulla sua linea di occupato/pronto, per indicare che è alla ricerca di dati. Lo stato logico di "occupato", viene mantenuto dalla periferica fino a che il dato non è disponibile. Lo stato logico cambia poi per indicare che il dato è "pronto". Durante la conversione dati o il periodo di raccolta, il microcomputer controlla la linea di occupato/pronto della periferica, in attesa del cambiamento logico che indica che i dati sono pronti. Quando si rileva il cambiamento, quindi, il microcomputer inserisce i dati dalla periferica. Tale procedura di I/O è illustrata in Fig. 5-1A.

Lo scambio di informazioni fra un microcomputer e una periferica, sotto forma d'impulsi per sincronizzare il trasferimento dati, è analogo allo scambio di saluti fra due persone che stringono l'una la mano all'altra. A causa di questa similitudine, l'uso di flag di strobe occupato/dati, nel trasferimento condizionato di dati, è spesso chiamato *handshaking* e la tecnica descritta come, *handshaking di I/O*.

5-2 CONFRONTO FRA HANDSHAKING DI I/O A COMANDO DI STATO E A COMANDO DI INTERRUPT

Si possono individuare *due tipi* di handshaking di I/O, che sono illustrati in Fig. 5-1. Il primo tipo è *l'handshaking di I/O a comando di stato*, che è stato descritto sopra e fa da supporto ad un'interfaccia OCCUPATO/STROBE DATI. Il problema di questo tipo di handshaking di I/O è che il microcomputer deve attendere, in un loop di ritardo, che la periferica abbia completato il proprio trasferimento di dati, o conversione dati (vedi Fig. 5-1A). Mentre ciò è accettabile in alcuni sistemi a microcomputer dedicati, che hanno un piccolo numero di periferiche, in sistemi più grandi, quei ritardi, che possono superare 1 ms (millisecondo), possono essere intollerabili, dal momento che possono impedire al microcomputer di assolvere altri compiti di elaborazione dati.

Il secondo tipo di handshaking di I/O, è *l'handshaking di I/O a comando di interrupt*. Questo permette di superare il problema del tempo di ritardo, consentendo al microcomputer di procedere con i propri impegni di altre elaborazioni, fino a che una periferica sia pronta a trasferire i dati. In quell'istante l'interfaccia genera un impulso di interrupt, che va alla linea di interrupt del microcomputer, come illustrato in Fig. 5-1B. Tale impulso causa l'interruzione da parte del microcomputer, della propria attività e il servizio della causa di interrupt, cioè della periferica. In questo modo il microcomputer, è impegnato al servizio della periferica per un breve periodo, quando il servizio è realmente necessario. Il discorso interrupt e servizio di interrupt, è difficile e bisogna porre attenzione nell'assicurare che:

(i) Il programma principale non sia interrotto durante l'esecuzione di codici essenziali.
(ii) La routine di servizio giusta, sia velocemente e facilmente individuata.
(iii) Lo stato del microcomputer all'istante dell'interrupt, sia conservato durante il servizio dell'interrupt.
(iv) Il sistema non deve diventare limitato dall'interrupt, cosicchè spende tutto il proprio tempo nel servizio di dispositivi interrompenti.

Il PPI può essere usato sia in modo 0 che 1, per handshaking di I/O unidirezionale. Il funzionamento in modo 0, combinato con la caratteristica di set/reset bit del PPI, può essere usato per implementare, in maniera facile, un'interfaccia handshaking I/O a comando di stato. Il funzionamento in modo 1, è stato progettato per handshaking di I/O a comando di interrupt e nel Capitolo 6 si discuterà tale tecnica.

5-3 IMPLEMENTAZIONE DELL'HANDSHAKING DI I/O A COMANDO DI STATO CON IL PPI

(A) Hardware

La tecnica del handshaking di I/O a comando di stato, illustrata in Fig. 5-1A, può essere facilmente implementata in un microcomputer basato sull'8080 mediante:

(i) L'impiego del PPI, funzionante in modo 0 per l'ingresso e l'uscita di dati e per l'ingresso dello stato di pronto/occupato della periferica.

(ii) L'impiego della caratteristica di set/reset bit della porta C del PPI, per generare gli impulsi di strobe dati.

Dal momento che molti dispositivi periferici, sono progettati per l'I/O a comando di stato, il PPI è spesso usato per implementare questo tipo di tecnica di handshaking, particolarmente, dove i ritardi conseguenti non sono importanti.

Si possono individuare diverse importanti caratteristiche del handshaking di I/O a comando di stato, esse sono illustrate in Fig. 5-1A:

- L'impulso di *strobe dati*, sia per l'ingresso che per l'uscita di dati, è *sempre* generato dal microcomputer ed applicato alla periferica.
- Il flag di *stato pronto/occupato* è sempre generato dalla periferica e deve essere letto dal microcomputer.
- I dati da inviare in uscita devono subire latch ed essere trattenuti dal microcomputer, fino a che la periferica sia pronta a riceverli. Nel funzionamento del PPI in modo 0, subiscono latch, i dati in uscita dalle porte A, B e C.
- L'ingresso di dati avverrà di solito con latch e saranno trattenuti dalla periferica per qualche istante, dopo del quale si indicherà che i dati sono pronti.
- Si ha una gran quantità di interazioni fra CPU e periferica durante il trasferimento dati, ciò a causa della necessità da parte del processore di controllare regolarmente la linea di stato della periferica.

La Fig. 5-2, indica i passi necessari alla lettura da una periferica ed alla scrittura di dati alla medesima, impiegando un handshaking d'interfaccia I/O a comando di stato, basato sul PPI. In generale, le linee dati della periferica, sono collegate alla porta A o alla B, la quale sarà poi programmata impiegando la parola di controllo del modo, per l'ingresso o l'uscita in modo 0, a seconda dei casi. L'autore consiglia, per convenzione standard, di usare i bit superiori della porta C, PC7 - PC4, come linee di strobe dati e i bit inferiori della porta C, PC3 - PC0, per ricevere le linee di stato della periferica. La porta C superiore, dovrà quindi essere programmata quale uscita in modo 0 e la porta C inferiore come ingresso in modo 0. La Fig. 5-3 illustra questo approccio di assegnazione delle porte per il caso di un perforatore di nastro di carta ed un lettore di nastro di carta, interfacciati al PPI con handshaking di I/O a comando di stato.

(B) Software

Diamo ora uno sguardo alle subroutine che potranno essere usate rispettivamente, per ottenere dati dal lettore di nastro di carta e per inviare dati al perforatore. Il primo passo, ad ogni modo, come al solito, è l'inizializzazione del PPI. In questo caso la porta A e la C superiore, devono essere programmate come uscite in modo 0, mentre la porta B e la C inferiore devono essere configurate per l'ingresso in modo 0. Facendo riferimento al formato della parola di controllo riportata in Fig. 2-2A, la parola di controllo del modo è 1 0000 011, oppure 203.

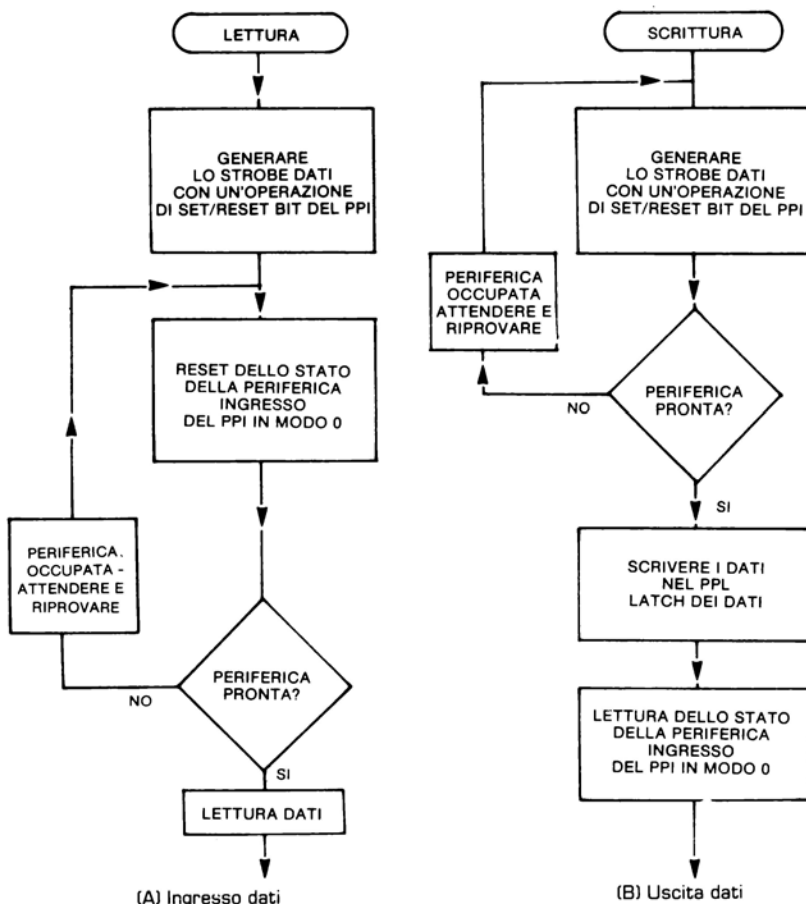



Fig. 5-2. Diagrammi di flusso delle procedure necessarie per inserire e far uscire dati, attraverso il PPI, impiegando handshaking di I/O a comando di stato.

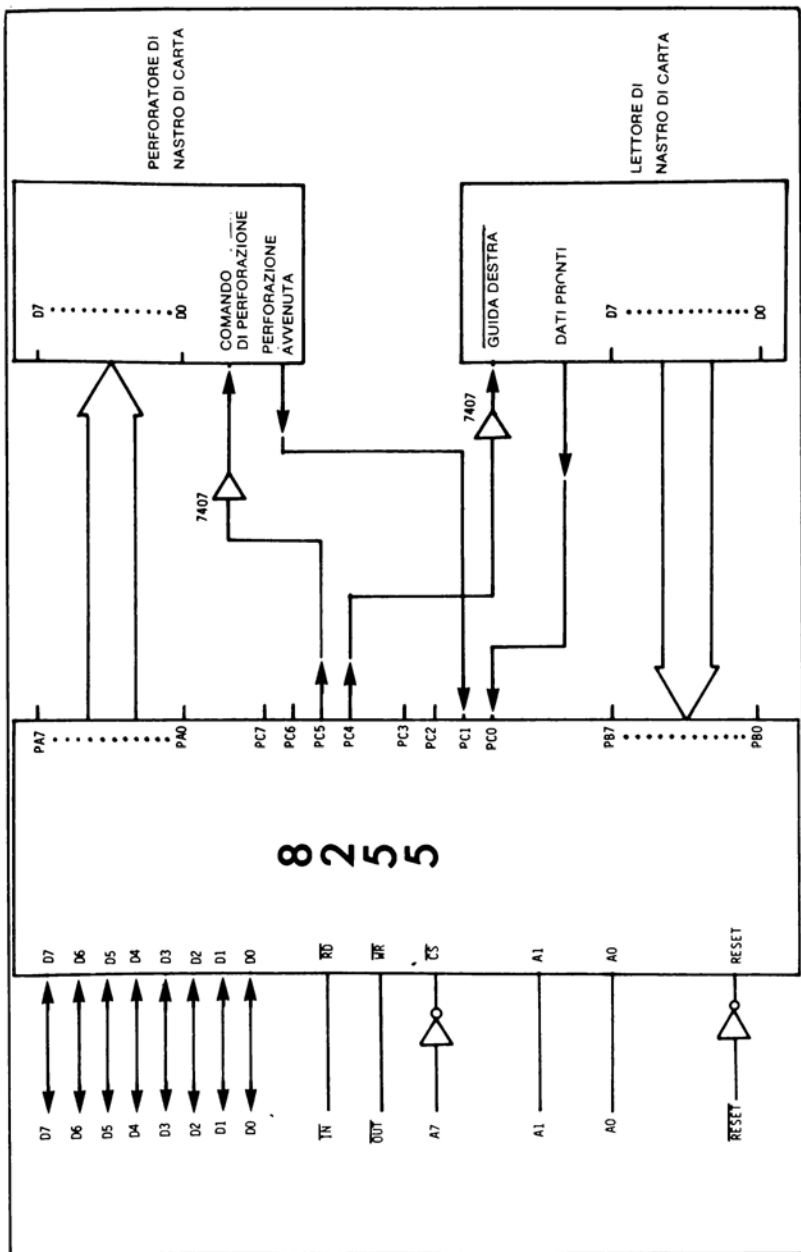



Fig. 5-3. Assegnazione delle porte del PPI per l'interfacciamento di un lettore e perforatore di nastro di carta con handshaking di I/O a comando di stato.

Come negli esperimenti dei precedenti Capitoli, dove le linee di controllo del PPI $\overline{CS}$, A_1 e A_0 erano collegate allo stesso modo, i codici dispositivi del PPI, per le porte da A fino a C e per il registro di controllo sono rispettivamente 200, 201, 202 e 203.

Programma 5-1 (Fig. 5-4)

I diagrammi di flusso delle subroutine per lettura e scrittura per nastro di carta, sono riportate, in forma generalizzata, in Fig. 5-2. Le subroutine del perforatore di carta, sono state scritte per fornire un esempio delle tecniche di programmazione impiegate per l'handshaking a comando di stato.
Il comando di perforazione inviato al perforatore di nastro di carta, è un segnale attivo alto, che fa muovere il nastro ed avvia la perforazione. Tale segnale altro non è che il segnale di *strobe dati* del perforatore di nastro di carta, ed è collegato al bit PC5 della porta C. Le parole di controllo di set/reset bit necessarie per mettere a uno oppure a zero il bit PC5, quindi per il set o reset dell'ingresso di controllo del perforatore, possono essere costruite in base alla Fig. 2-2B e sono riportate in tabella 5-1.

```
/
/QUESTO PROGRAMMA INIZIALIZZA IL PPI, INSERISCE I DATI DAL
/LETTORE DEL NASTRO DI CARTA E LI INVIA AL PERFORATORE
/DEL NASTRO.
            MVIA    /CARICARE IN A LA PAROLA DI CONTROLLO
            MODE    /DEL MODO DEL PPI
            OUT     /INVIARE LA PAROLA DI CONTROLLO DEL
                    /MODO AL
            CNTRL   /REGISTRO DI CONTROLLO DEL PPI
            CALL    /INSERIRE UN BYTE DI DATI DAL
            TAPEIN  /LETTORE DEL NASTRO DI CARTA
            0
            CALL    /INVIARE IL BYTE DI DATI AL
            TAPOUT  /PERFORATORE DI NASTRO DI CARTA
            0
            HLT     /HALT
```

Fig. 5-4. Programma 5-1.

L'uscita di perforatore pronto, è un segnale attivo alto che va al valore logico 1, per indicare che il perforatore è *pronto* a ricevere dati da mettere su nastro. Un valore logico basso su tale linea, che è collegata a PC1, indica che il perforatore è *occupato*. I codici della subroutine di perforazione, conseguenti al diagramma di flusso di Fig. 5-2B, sono riportati di seguito:

Tabella 5-1. Parola di Controllo Set/Reset Bit Necessarie per il Set e il Reset di PC5 e PC4		
Azione	**Simbolo**	**Parola di controllo di set/reset bit**
Impostazione Guida Destra (PC4)	PC4SET	1 000 100 1 = 211
Reset Guida Destra (PC4)	PC4RST	1 000 100 0 = 210
Impostazione comando perforazione (PC5)	PC5SET	1 000 101 1 = 213
Reset comando perforazione (PC5)	PC5RST	1 000 101 0 = 212

Subroutine di Trasmissione (Fig. 5-5)

Questo programma illustra la tecnica standard di mascheratura, impiegata per determinare lo stato di una periferica. In questo caso lo stato logico dei bit della porta C, è determinato da un'istruzione d'ingresso della porta C. Lo stato logico del bit PC1, che è collegato alla linea "perforatore pronto", viene poi determinato mediante un AND logico fra il contenuto dell'accumulatore ed un byte di maschera, i cui bit sono tutti posti a 0 tranne il bit D1, che viene messo al valore logico 1. Se il bit PC1 è al valore logico 0 (perforatore occupato), il risultato dell'operazione logica è zero. L'istruzione "salta se il risultato è zero" (JZ),

```
SUBROUTINE TAPOUT
        /DESCRIZIONE: QUESTA SUBROUTINE INVIA UN BYTE
        /             DI DATI DALL'ACCUMULATORE AL
        /             PERFORATORE DI NASTRO DI CARTA
                MOVBA  /SALVARE IL BYTE CHE DEVE ESSERE
                       /TRASMESSO
        LOOP,   IN     /PRELEVARE LO STATO DEL PERFORATORE
                PORTC  /IL PERFORATORE È OCCUPATO?
                ANI    /IL PERFORATORE È OCCUPATO
                PC1MSK /BYTE MASCHERA 00 000 010
                JZ     /PC1 = "0". PERFORATORE OCCUPATO
                       /QUINDI
                LOOP   /ANDARE A LOOP E PROVARE DI NUOVO
                0
                MOVAB  /PC1 = "1". PERFORATORE PRONTO QUINDI
                       /RIPRISTINARE IL BYTE CHE DEVE ESSERE
                       /TRASMESSO
                OUT    /INVIARE IN USCITA IL BYTE ALLA PORTA A
                PORTA
                MVIA   /CARICARE IN A IL BYTE DI CONTROLLO DI
                PC5SET /"SET STROBE DATI"
                OUT    /SET STROBE DATI
                CNTRL
                DCRA   /GENERAZIONE DEL BYTE DI "RESET STROBE
                       /DATI"
                OUT    /RESET STROBE DATI
                CNTRL
                RET    /RITORNO
```

Fig. 5-5. Subroutine TAPOUT o di trasmissione.

costringerà il programma a riciclare, cosicchè esso inserirà nuovamente lo stato della porta C per il controllo. Se il bit PC1 è al valore logico 1 (perforatore pronto), il risultato dell'operazione logica non è zero ed il programma non ricicla, inizia invece ad inviare in uscita al perforatore un byte attraverso la porta A.
Tale tecnica di controllo di un flag di stato è generale e può essere usata per la subroutine del lettore di nastro. Bisogna fare attenzione nella scelta fra le istruzioni JNZ e JZ da usare nel programma. Tale scelta dipende dall'interpretazione dello stato logico del flag occupato/pronto della periferica. Nel lettore del nastro di carta, la linea di stato "Dati Pronti" è di nuovo un flag attivo alto con un valore logico 1, per indicare che i dati sono pronti ed un valore logico 0, per indicare che il lettore è occupato. Da qui segue che si dovrà usare nuovamente un'istruzione JZ con il byte di maschera 00 000 001.
Un esercizio utile è quello di scrivere la subroutine per il lettore del nastro di carta. Si ricordi, che la generazione di impulsi di strobe, che sono normalmente alti o normalmente bassi, si può velocemente ottenere via software sfruttando la possibilità di set/reset bit del PPI. Ciò è stato esaminato nel Capitolo 4. Anche "l'inversione" del rilevamento di un flag, può essere facilmente ottenuto via software, mediante la scelta adeguata dell'istruzione JNZ oppure JZ.

5-4 SOMMARIO DELL'ESPERIMENTO 5-1

Esperimento	*Descrizione*
5-1	Lo scopo dell'esperimento, è quello di illustrare la tecnica di handshaking di I/O a comando di stato, impiegando il PPI nel funzionamento in modo 0. L'esperimento si divide in due parti: (a) Operazione d'ingresso a comando di stato. (b) Operazione d'uscita a comando di stato. Si fornisce un circuito che può essere usato per ambedue le parti dell'esperimento e si dà un programma per l'operazione di ingresso. Si propone inoltre, di scrivere una subroutine per inviare in uscita i dati, impiegando handshaking di I/O a comando di stato. Alla fine dell'esperimento è riportata una subroutine per un confronto.

ESPERIMENTO 5-1
HANDSHAKING A COMANDO DI STATO IN INGRESSO ED USCITA

Scopo

Lo scopo di questo esperimento, è quello di illustrare la tecnica di *I/O a comando di stato*, impiegando il funzionamento in modo 0 del PPI sia per l'ingresso che per l'uscita; per la generazione degli impulsi di strobe, si sfrutta la possibilità di set/reset bit del PPI. L'esperimento si divide in due parti:

(a) Operazione d'ingresso a comando di stato.
(b) Operazione d'uscita a comando di stato.

Viene fornito un circuito per entrambe le parti e viene dato un programma per l'operazione d'ingresso. Si può provare a scrivere e a controllare una subroutine, per inviare in uscita i dati, impiegando handshaking di I/O a comando di stato. Le specifiche per questa subroutine sono date nel passo 8. La subroutine dell'autore è riportata, per confronto, alla fine dell'esperimento.

Ingresso a comando di stato

Passo 1

Collegare il circuito mostrato nello schema di Fig. 5-8. In questo circuito la porta B, è usata per inserire i dati provenienti dagli switch logici. La porta A è usata per inviare in uscita, dati, ad un paio di latch a 4 bit SN7475. La porta C superiore (PC7-PC4), è usata per generare impulsi di strobe attivi bassi, con il bit PC7 che abilita i latch attraverso un inverter (i latch vengono abilitati da un 1 logico) e con PC7 che fornisce l'impulso di strobe. In questa semplice simulazione di handshaking di I/O a comando di stato, l'impulso di strobe per l'ingresso dati, non è fisicamente necessario ed esso viene così usato per pilotare una lampada di rivelazione: La porta C inferiore, (PC3 - PC0) è stata configurata come ingresso per lo stato OCCUPATO/PRONTO del dispositivo collegato in ingresso alla porta B (PC2) e del dispositivo collegato in uscita alla porta A (PC3)

Programma (Fig. 5-6)

```
                /
                /PROGRAMMA PER I/O A COMANDO DI STATO
                /
                DB MODE 203
                DB CNTRL 203
                DB PC7SET 017
                DB PC6SET 015
                DB PC6RST 014
                *003 010
003 010 076              MVIA      /INIZIALIZZAZIONE DEL PPI. PORTA A E
                                   /PORTA C
003 011 203              MODE      /SUPERIORE: O/P. PORTA B E PORTA C
                                    /INFERIORE: I/P
003 012 323              OUT
003 013 203              CNTRL
003 014 076              MVIA      /SET DI PC7, STROBE ATTIVO
003 015 017              PC7SET    /BASSO PER LA PORTA A
003 016 323              OUT
003 017 203              CNTRL
003 020 076              MVIA      /SET DI PC6, STROBE ATTIVO
003 021 015              PC6SET    /BASSO PER LA PORTA B
003 022 323              OUT
003 023 203              CNTRL
003 024 061              LXISP     /CARICARE LO STACK
003 025 000              000       /POINTER
003 026 004              004
003 027 315     LOOP1,   CALL
003 030 042              READ
003 031 003              0
003 032 000              NOP
003 033 000              NOP
003 034 000              NOP
003 035 323              OUT       /INVIARE IL CONTENUTO
                                   /DELL'ACCUMULATORE
003 036 000              000       /ALLA PORTA 0
003 037 303              JMP
003 040 027              LOOP1
003 041 003              0
                /
                /SUBROUTINE: READ
                /DESCRIZIONE: QUESTA ROUTINE LEGGE I DATI DALLA
                /PORTA B IMPIEGANDO HANDSHAKING DI I/O A COMANDO DI
                STATO
                /
003 042 365     READ,    PUSHPSW   /SALVARE LA PAROLA DI STATO
                                   /DEL PROGRAMMA ED IL CONTENUTO
                                   /DELL'ACCUMULATORE
003 043 076              MVIA      /GENERARE UN IMPULSO DI
003 044 014              PC6RST    /STROBE ATTIVO BASSO
003 045 323              OUT
003 046 203              CNTRL
003 047 315              CALL      /GENERARE UN RITARDO IN MODO CHE
003 050 200              DELAY     /L'IMPULSO DI STROBE POSSA ESSERE VISTO
003 051 003              0
```

Fig. 5-6. Programma per l'Esperimento 5-1 (segue).

```
003 052 074          INRA
003 053 323          OUT
003 054 203          CNTRL
003 055 333   LOOP2, IN          /STATO OCCUPATO/PRONTO
                                 /DELL'INGRESSO DALLA
                                 /PORTA C
003 056 202          202
003 057 346          ANI
003 060 004          004         /BYTE MASCHERA PER IL BIT PC2 DELLA
                                 /PORTA C
003 061 302          JNZ         /PC2 = "1". QUINDI, LA PORTA B È
003 062 100          POINTX      /OCCUPATA, SALTARE ALL'INGRESSO DATI
003 063 003          0
003 064 016          MVIC        /PC2 = "0". QUINDI LA PORTA B È
003 065 005          005         /OCCUPATA, QUINDI GENERARE
003 066 015   LOOP3, DCRC        /UN RITARDO
003 067 315          CALL
003 070 200          DELAY
003 071 003          0
003 072 302          JNZ
003 073 066          LOOP3
003 074 003          0
003 075 303          JMP         /CONTROLLARE DI NUOVO LO STATO
003 076 055          LOOP2       /DELLA PORTA B
003 077 003          0
003 100 333   POINTX,IN
003 101 201          201
003 102 107          MOVBA       /SALVARE IL CONTENUTO DELLA PORTA B
003 103 361          POPPSW
003 104 311          RET

              /SUBROUTINE:       DELAY
              /DESCRIZIONE:      QUESTA SUBROUTINE GENERA UN RITARDO
              /                  DI UN SECONDO PER UN MICROCOMPUTER CHE
              /                  HA
              /                  UN CLOCK DI 750 KHZ

              *003 200
003 200 365   DELAY, PUSHPSW
003 201 325          PUSHD
003 202 021          LXID        /CARICARE I BYTE DI TEMPORIZZAZIONE PER
003 203 000          000         /IL RITARDO DI UN SECONDO
003 204 200          200
003 205 033   LOOP4, DCXD
003 206 173          MOVAE
003 207 262          ORAD
003 210 302          JNZ
003 211 205          LOOP4
003 212 003          0
003 213 321          POPD
003 214 361          POPPSW
003 215 311          RET
```

Fig. 5-6. Programma per l'Esperimento 5-1.

Schema del Circuito (Fig. 5-8)

Con i dati forniti e usando il formato di parola di controllo del modo di Fig. 2-2A, costruire la parola di controllo del modo necessaria per configurare il PPI come indicato in Fig. 5-7. Per costruire la parola di controllo set/reset bit, per il set e il reset dei bit PC7 e PC6 della porta C, impiegare il formato di parola di controllo set/reset bit di Fig. 2-2B.

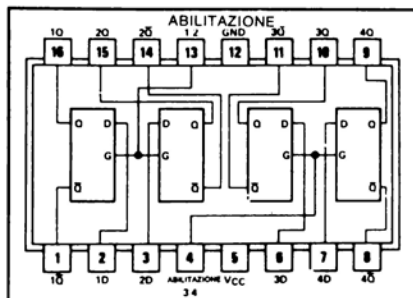
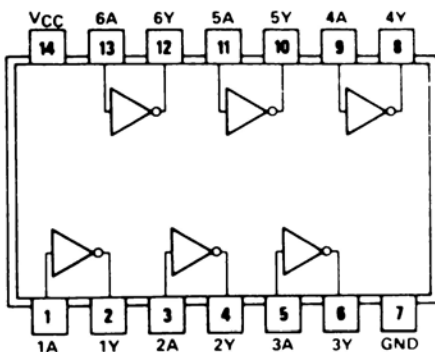
Configurazione dei pin del chip circuiti integrati (Fig. 5-7)

(A) 7475.

(B) 7404.

Fig. 5-7. Configurazione dei pin del chip IC.

Scrivere i risultati nello spazio fornito.

I risultati richiesti sono i seguenti:

Byte parola di controllo modo: 1 0000 011 oppure 203 (MODO)

Byte parola set/reset bit

PC7 set: 0 000 111 1 oppure 017 (PC7 SET)
PC7 reset: 0 000 111 0 oppure 016 (PC6 RST)
PC6 set: 0 000 110 1 oppure 015 (PC6 SET)
PC6 reset: 0 000 110 0 oppure 014 (PC6 RST)

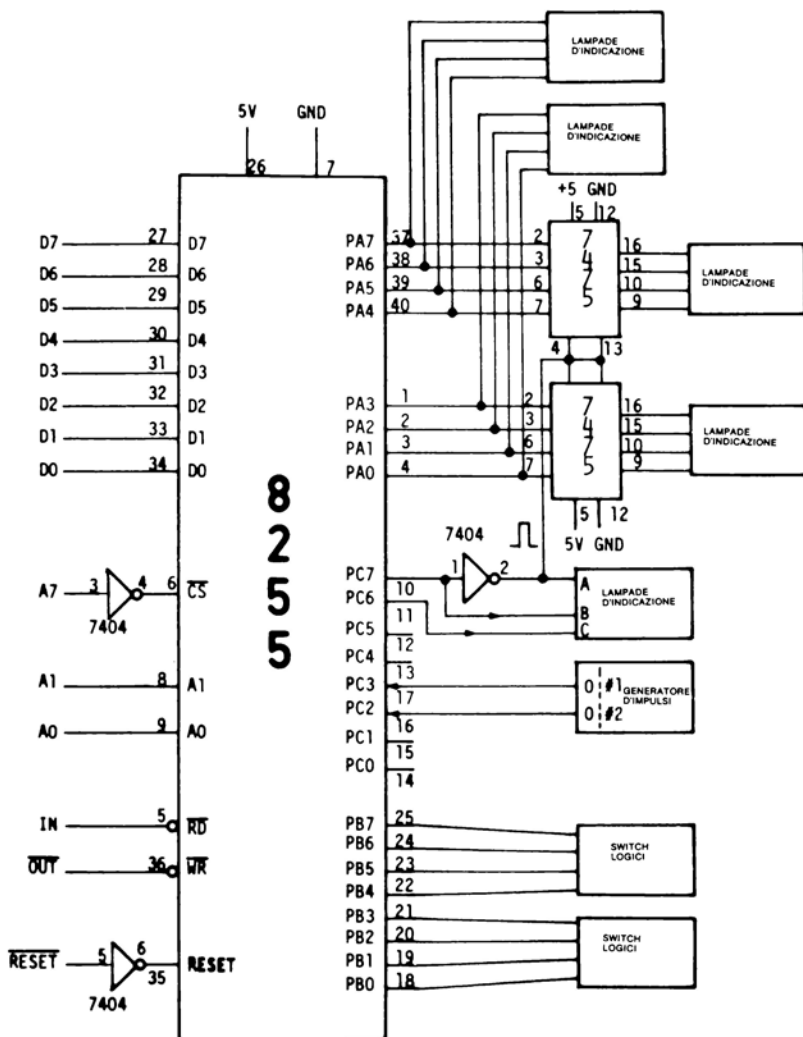



Fig. 5-8. Circuito per l'Esperimento 5-1.

Notare come tali byte, vengono impiegati nel programma per inizializzare il PPI e generare l'impulso di strobe della porta A.

Passo 2

Caricare il programma in memoria. Studiare il listing del programma. Si può vedere che esso è diviso in una parte di programma principale ed in una subroutine di lettura della porta B. In questa prima parte dell'esperimento, si tratterà solamente la lettura a comando di stato. Nello spazio sotto fare uno schizzo del flowchart del programma. Tale flowchart della subroutine di lettura, dovrà essere simile a quello dato in Fig. 5-2A. Annotare, nello spazio sotto, ogni differenza trovata.

Le due differenze consistono nei dati inseriti attraverso la porta B che sono:

(i) uscita alla porta 0 e
(ii) protezione nel registro B dell'8080A.

Passo 3

Porre gli switch logici a 1010 1010. Eseguire il programma, iniziando alla locazione 003 010, osservando nel frattempo l'uscita alla porta 0 e le lampade di rivelazione che sono collegate a PC7 e PC6. Decrivere e spiegare cosa si osserva.

Abbiamo visto che la lampada collegata al bit PC6, della porta C, si è spenta per circa un secondo. Non c'è stato alcun cambiamento nel contenuto della porta 0. Il programma è entrato nella subroutine READ che genera un impulso attivo basso di 1 secondo sul bit PC6 della porta C. Ora sta avvenendo il controllo del bit PC2 per lo stato occupato /pronto della porta B.

Passo 4

Avviare ora, il generatore d'impulsi B per almeno 3 secondi, poi bloccarlo. Cosa si osserva alla porta 0?

Abbiamo visto che il byte dati, 252, impostato sugli switch logici, è stato portato fuori sulla porta 0. Dal momento che il generatore d'impulsi B, è il flag di stato OCCUPATO/PRONTO della porta B, quali sono le conclusioni che si possono trarre a proposito degli stati logici che rappresentano lo stato OCCUPATO e lo stato PRONTO della "periferica" alla porta B?

Poichè il generatore d'impulsi, è normalmente nello stato di 0 logico e diventa 1 logico solamente quando viene avviato, il programma interpreta un 1 logico come uno stato PRONTO della porta B.

Passo 5

Impostare sugli switch il valore 1111 0000. Avviare ora e bloccare il generatore d'impulsi B il più velocemente possibile. Si osserva qualche cambiamento nel byte visualizzato alla porta 0? Se no, perchè?

Noi, il più delle volte, non abbiamo trovato alcun cambiamento nel byte visualizzato alla porta 0. La ragione di ciò è che il programma non visualizza il flag di stato della porta B, durante l'intervallo di tempo in cui il generatore d'impulsi resta azionato. Esso, infatti, sta riciclando sulle istruzioni all'interno di LOOP3 del programma, che generano un ritardo di 4 secondi.
La visualizzazione di un flag di stato, è conosciuta come "polling" e tale step, è stato aggiunto per illustrare un problema potenziale che può sorgere se il flag di stato non è controllato abbastanza spesso. A questo punto, togliere l'istruzione di chiamata alla locazione 003 067, inserendo delle NOP (000) nelle locazioni di memoria 067, 070 e 071 (L0 byte indirizzo). Ciò ridurrà virtualmente a zero il ritardo. Modificare ora gli switch logici ed avviare e bloccare il generatore d'impulsi B. Ripetere l'operazione con diversi byte sugli switch logici. Si osserva che il byte alla porta B viene letto ogni volta? Dovrebbe essere così.

Passo 6

Avviare e bloccare il generatore d'impulsi A. Che cosa si vede, se si vede qualcosa? Perchè?

Non vediamo alcun cambiamento alla porta 0. Il generatore d'impulsi A, rappresenta il flag di stato OCCUPATO/PRONTO della porta A, la quale è stata programmata come uscita dati. Essa non è controllata dalla subroutine READ.

Passo 7

Inserire nelle L0 locazioni di memoria 032 e 033 i seguenti byte:

```
003 032 326 SUI
003 033 060 060
```

Impostare gli switch logici a 066 (ASCII per il decimale 6). Eseguire il programma ed avviare e bloccare il generatore d'impulsi B. Quale byte viene visualizzato alla porta 0 ? Perchè ?

Alla porta 0, quando il programma viene eseguito ed il generatore d'impulsi è azionato e bloccato, viene visualizzato il byte 006. La subroutine READ, preleva il byte 066 dalla porta B, quando il generatore d'impulsi viene azionato. Questo è il valore dell'accumulatore, quando l'istruzione, che abbiamo inserito sopra, viene eseguita in seguito al ritorno dalla subroutine READ. Poichè 006 è il valore decimale 6, in quale modo il microcomputer ha elaborato i dati provenienti dalla porta B?

Il microcomputer ha eseguito una conversione da ASCII a BCD. Tale tecnica è molto utile, dal momento che i codici ASCII per i numeri decimali da 0 a 9, vanno rispettivamente da 060 fino a 071. A questo punto, impostare il codice ASCII per 8 (070) sugli switch logici e si otterrà, dal microcomputer, la conversione in BCD, sulla porta 0, avviando e bloccando il generatore d'impulsi. Avviene ciò? Per noi è così.

Uscita a comando di stato

Passo 8

Siete ora invitati a scrivere una subroutine, WRITE, impiegando il flowchart di Fig. 5-2B. La subroutine serve per inviare in uscita i dati, *memorizzati nel registro B*, sulla porta A, impiegando il PPI in funzionamento in modo 0 e a comando di stato. Il programma dovrà iniziare alla locazione 003 120. Il bit PC7 (vedere lo schema del circuito), è uno strobe

```
                DB CNTRL 203
                DB PC7RST 017
                *003 120
                /
                /SUBROUTINE: WRITE
                /DESCRIZIONE: QUESTA SUBROUTINE INVIA IN USCITA
                /I DATI CONTENUTI NEL REGISTRO B, ALLA PORTA A
                /IMPIEGANDO HANDSHAKING DI I/O A COMANDO DI STATO.
                /
003 120 365     WRITE, PUSHPSW
003 121 333     LOOP2, IN       /STATO OCCUPATO/PRONTO
                                /DELL'INGRESSO
003 122 202            202      /DALLA PORTA C
003 123 346            ANI
003 124 010            010      /BYTE MASCHERA PER IL BIT PC3 DELLA
                                /PORTA C
003 125 312            JZ       /PC3 = "0". PORTA A OCCUPATA, PROVARE
003 126 121            LOOP2    /DI NUOVO
003 127 003            0
003 130 170            MOVAB    /PORTA A PRONTA, RIPRISTINARE I DATI DA
                                /INVIARE IN USCITA
003 131 323            OUT      /INVIARE IN USCITA I DATI ALLA PORTA A
003 132 200            200
003 133 076            MVIA     /GENERARE UN IMPULSO DI STROBE
                                /ATTIVO BASSO
003 134 017            PC7RST
003 135 323            OUT
003 136 203            CNTRL
003 137 074            INRA
003 140 323            OUT
003 141 203            CNTRL
003 142 361            POPPSW
003 143 311            RET
```

Fig. 5-9. Subroutine WRITE.

attivo basso ed il flag di stato OCCUPATO/PRONTO della porta A, deve essere inserito sul bit PC3 della porta C, con la convenzione che un 1 logico rappresenta lo stato PRONTO. Dal momento che il contenuto dell'accumulatore e lo stato logico del flag zero, verranno distrutti durante l'esecuzione della subroutine WRITE, non dimenticarsi di caricare la parola di stato del programma (PUSH PSW) nello stack all'inizio della routine e di ripristinarla immediatamente prima del ritorno allo stato iniziale. (Vedere la Fig. 5-9 per il confronto).

Avendo scritto la subroutine si può incorporarla nel programma, inserendo un'istruzione CALL alla locazione 003 032, come di seguito:

```
003 032 315 CALL
003 033 120 WRITE
003 034 003   0
```

Se il programma comincia ora ad essere eseguito dalla locazione 003 010, i dati saranno inseriti dalla porta B e usciranno dalla porta A, quando verranno avviati rispettivamente i generatori di impulsi B e A. I dati che erano visualizzati sulla porta 0 nei passi da 3 a 7, possono ora essere visualizzati sulle lampade di rivelazione della porta A del PPI, azionando il generatore d'impulsi A.

CAPITOLO 6

HANDSHAKING DI I/O A COMANDO DI INTERRUPT: FUNZIONAMENTO DEL PPI IN MODO 1

6-1. INTRODUZIONE

Il funzionamento in modo 1 del PPI, è stato progettato per permettere trasferimenti condizionati di dati fra l'8080A e le sue periferiche, senza che il microcomputer debba continuamente controllare lo stato delle periferiche, come descritto nel Capitolo 5. Il polling dello stato delle periferiche è, chiaramente, un modo poco efficiente di sfruttare, in molti casi, le possibilità dell'8080A. La regola di funzionamento del PPI in modo 1, è quella di *regolare* il trasferimento dati, microcomputer periferica, impiegando segnali di handshaking ed *interrompere* il microcomputer solamente quando è assolutamente necessario. Il microcomputer è così libero per altri compiti di sequenze logiche e di elaborazione dati.

La Fig. 6-1 illustra, in modo schematico, come possa essere usato l'8255, funzionante in modo 1, per l'ingresso e l'uscita dati. In questo esempio il dispositivo d'ingresso è una tastiera completamente decodificata ed il dispositivo d'uscita, è un video display.

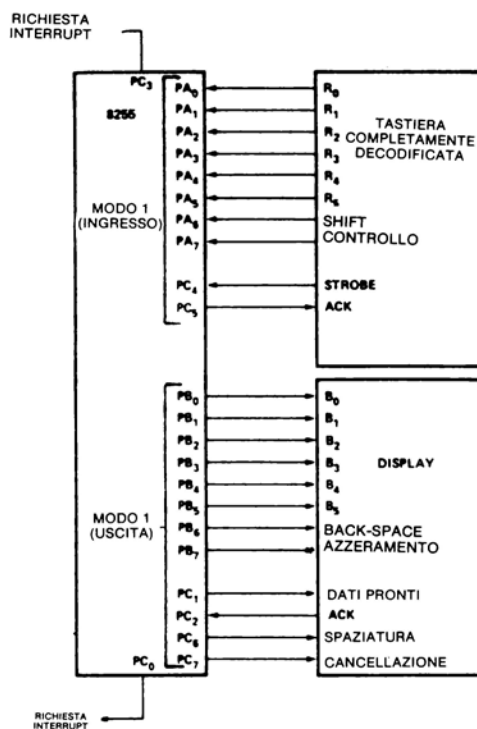
La caratteristica importante di questa interfaccia in modo 1, che la distingue dall'interfaccia a comando di stato con strobe occupato/pronto di Fig. 5-3, è l'*impiego di interrupt* per segnalare al microcomputer che si è ultimato un trasferimento dati fra PPI e le periferiche.
Anche i segnali di handshaking, sono leggermente diversi da quelli usati in Fig. 5-3. In Fig. 6-1, vengono usati segnali di strobe/accettazione. Il segnale di *strobe dati*, è generato dal dispositivo da cui i dati hanno origine. Il dispositivo ricevente segnala poi, al trasmettitore, che il dato è stato accettato, alzando il flag di *accettazione* (ACK per entrambe le periferiche di Fig. 6-1). Quando si preme un tasto, ed il codice corrispon-

Per gentile concessione della Intel Corp.

Fig. 6-1. Esempio d'impiego del PPI in funzionamento in modo 1 per l'ingresso e l'uscita dei dati.

dente è disponibile sulle linee dati da R0 a R5, per l'ingresso nel microcomputer, la tastiera genera un segnale di "strobe" che il PPI accetta quando il microcomputer ha letto i dati dalla porta A. Allo stesso modo, quando il PPI ha un byte di dati pronto alla porta A, per l'uscita verso il video display, esso genera un impulso sulla linea "dato pronto" del video display, per segnalargli che il dato è disponibile. Quando il display ha ricevuto il byte, riconosce la sua ricezione con un impulso sulla linea ACK. Contrariamente a quanto sopra, i due segnali di strobe dati di Fig. 5-3, erano generati dal PPI, per richiedere trasferimento dati alle periferiche.

La differenza fra i segnali di handshaking di strobe occupato/dati e strobe dati/accettazione, è quindi essenzialmente quella del luogo in cui nasce *l'iniziativa* per il trasferimento dati. In un'interfaccia basata sullo strobe occupato/dati, l'iniziativa parte dal microcomputer, sia per l'in-

gresso che per l'uscita. In un'interfaccia basata sullo strobe dati/accettazione, l'iniziativa parte sempre dal dispositivo che genera i dati. Questo può essere il microcomputer (attraverso il PPI) o la periferica, come illustrato in Fig. 5-1B.
Il motivo di ciò, è che i segnali di handshaking di strobe dati/accettazione, sono più convenienti per il funzionamento in modo 1 del PPI, dove quest'ultimo dirige il trasferimento di dati indipendentemente dal microcomputer. Consideriamo ora più in dettaglio le caratteristiche del PPI, programmato per il funzionamento in modo 1.

6-2. CARATTERISTICHE DEL PPI IN MODO 1

L'assegnazione delle 24 linee d'interfaccia del PPI, per il funzionamento in modo 1, è riportata schematicamente in Fig. 6-2. Le porte A e B, sono usate per l'handshaking di I/O unidirezionale e la porta C fornisce le linee di controllo necessarie. Ogni porta (A o B) in modo 1, è costituita da una porta dati ad 8 bit, tre linee di controllo ed un po' di logica interna di supporto per l'interrupt. In questo modo restano libere due linee della porta C, che sono disponibili per I/O in modo 0, o per la generazione d'impulsi per mezzo della caratteristica di set/reset bit del PPI. Ogni porta dati ad 8 bit (porta A o B), può essere usata sia per operazioni d'ingresso che per operazioni d'uscita ed entrambe avvengono tramite latch. In Fig. 6-2, le linee dati e le linee di controllo o handshaking, associate ad ogni porta in modo 1, sono state messe fra parentesi. Notare che le due linee di I/O libere, sono state raggruppate con le linee di controllo della porta A. La ragione di ciò è che, nel funzionamento del PPI in modo 2, la porta A è usata in handshaking di I/O bidirezionale e impiega tutti i cinque bit della porta C (da PC7 a PC3) come linee di controllo. Si noti anche che l'assegnazione delle linee di controllo della porta A e della B, cambiano per l'ingresso e l'uscita. Consideriamo prima le linee di controllo per l'ingresso, per vedere come vengono usate per il controllo delle operazioni d'ingresso in modo 1.

(A) Ingresso in Modo 1

I segnali di controllo interno ed handshaking esterno, usati per controllare le operazioni d'ingresso in modo 1 alle porte A e B, sono illustrati in Fig. 6-3. Ogni porta ha tre segnali di handshaking esterni ($\overline{STB}$, IBF e INTR) ed una linea di controllo interna (INTE). Essi sono definiti come segue:

Fig. 6-2. Assegnazione delle linee d'interfaccia per il PPI in modo 1.

- $\overline{STB}$ *(Strobe d'Ingresso)*. Esso è il segnale di *strobe dati*, il quale indica che la periferica ha dei dati pronti per l'ingresso. Deve essere generato da una periferica esterna. Vengono usati i bit PC4 e PC2, uno 0 logico su questi ingressi, caricherà i dati nei latch d'ingresso rispettivamente della porta A o della porta B (cf PC4 in Fig. 6-1).

Fig. 6-3. Illustrazione dei segnali di controllo interni ed esterni, necessari per dirigere le operazioni d'ingresso in modo 1.

- *IBF (Buffer d'Ingresso Pieno)*. Questo è il segnale di *accettazione* del PPI, per i dati che sono stati messi in ingresso al PPI da una periferica. Un 1 logico su questa uscita, indica che i dati sono stati caricati nel latch d'ingresso della porta A (PC5) o della porta B (PC1). La linea *è* posta al valore logico 1, circa 300 ns dopo che l'ingresso $\overline{STB}$ sia andato al valore logico 0.
- *INTR (Richiesta Interrupt)*. Come il nome stesso suggerisce, questo segnale può essere usato per interrompere l'8080A quando i dati, provenienti da una periferica esterna, sono stati caricati nei latch della porta A o B, per essere inseriti nell'8080A. Questa uscita di richiesta interrupt, è posta al valore logico 1 se $\overline{STB}$, IBF e INTE, linee di controllo interno, sono tutte al valore logico 1.
- *INTE (Abilitazione Interrupt)*. Questo è un flip-flop interno di controllo interrupt, che può essere usato per inibire (INTE = 0) o abilitare (INTE = 1) la generazione di interrupt, sia da parte della porta A che da parte della porta B. Lo stato logico di INTE, è controllato usando la possibilità di set/reset bit della porta C. Il flip-flop di abilitazione interrupt della porta A, INTE A, è controllato tramite il set ed il reset di PC4, mentre INTE B, flip-flop di abilitazione della porta B, è controllato tramite il set ed il reset di PC2. Si noti che, nel funzionamento in modo 1, le operazioni di set/reset bit su PC4 e PC2 per controllare i flip-flop INTE, rispettivamente della porta A e della porta B, sono operazioni interne del PPI e non hanno effetto sugli stati logici dei pin PC4 e PC2 del PPI, i quali per l'ingresso dati in modo 1, vengono usati come linee di *strobe d'ingresso* ($\overline{STB}$). Il vantaggio dei flip-flop interni INTE del PPI, è che permettono all'utilizzatore del microcomputer, di disabilitare selettivamente delle periferiche sotto il controllo del software. Questa è una possibilità estremamente utile per avere disponibile il microcomputer in circostantze ben definite, ad esempio dove si sa che, in certi istanti, sarà richiesta tutta la potenza di elaborazione per un compito importante (elaborazione dati, I/O ecc.). La priorità di accesso alle risorse del sistema, può essere spostata in questo modo, per risolvere il picco di domande.

La Fig. 6-4, mostra un diagramma di temporizzazione per l'ingresso in modo 1, che può essere usato per fare una valutazione della sequenza di funzionamento per l'ingresso in modo 1. Le operazioni d'ingresso dati in modo 1, cominciano con un'impulso d'ingresso di *strobe*, attivo basso ($\overline{STB}$), proveniente da una periferica, il quale fa sì che la linea d'uscita di *buffer d'ingresso pieno* (IBF), vada al valore logico 1. Tale segnale può

essere usato come un segnale di accettazione del PPI. Il valore logico 1 indica che i dati sono stati caricati nel latch d'ingresso della porta, ma non sono ancora stati letti dal microcomputer. Lo step successivo dipende dallo stato del flip flop di abilitazione interrupt (INTE) della porta. Se questo è stato precedentemente posto a 1, impiegando la

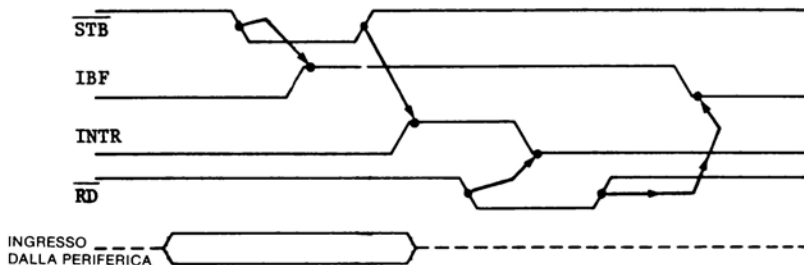



Fig. 6-4. Diagramma di temporizzazione per l'ingresso in modo 1. Si suppone che INTE sia al valore logico 1.

possibilità di set/reset bit della porta C, la linea di richiesta interrupt del PPI (INTR), sarà posta al valore logico 1 quando $\overline{STB}$ ritorna ad 1. Se INTE era al valore logico 0, INTR non sarà posto ad 1 quando $\overline{STB}$ andrà ad 1 e il dato proveniente dalla periferica verrà effettivamente perso. Supponendo che INTE fosse ad uno, il valore 1 logico risultante sulla linea di richiesta interrupt (INTR) può essere usato per interrompere il microcomputer. Il microcomputer deve poi determinare quale porta lo ha interrotto, usando un interrupt vettorizzato o per mezzo di polling. Dopo aver fatto ciò, esso legge il dato che è trattenuto nel latch d'ingresso della porta del PPI. Lo 0 logico che si genera sulla linea $\overline{RD}$ durante l'operazione di lettura del microcomputer, riporta INTR al valore logico 0. La linea IBF, viene posta a 0 dal fronte d'uscita di $\overline{RD}$ e da questo momento il PPI segnala alla periferica che il microcomputer ha ricevuto il dato. L'operazione d'ingresso in modo 1 è così completa.

Il vantaggio di questa procedura d'ingresso, è che un dispositivo d'ingresso può richiedere servizio dal microcomputer, semplicemente introducendo i propri dati nella porta A o B. Il PPI dirige le operazioni d'ingresso ed il microcomputer viene interrotto solamente quando il dato è realmente pronto per essere inserito dal PPI.

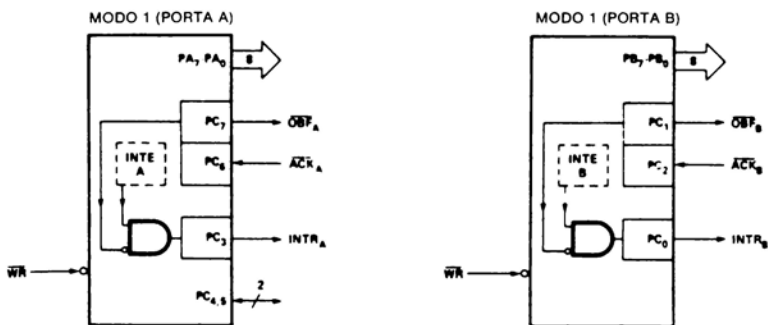



Fig. 6-5. Illustrazione dei segnali di controllo interni ed esterni, necessari per dirigere le operazioni di uscita in modo 1.

(B) Uscita in Modo 1

I segnali di controllo interno e di handshaking esterno, usati per il controllo delle operazioni d'uscita in modo 1, sono illustrati in Fig. 6-5. Per regolare le operazioni d'uscita, alle porte A e B, in modo 1, vengono impiegati tre segnali di handshaking esterno ($\overline{OBF}$, $\overline{ACK}$ e INTR) ed il segnale di controllo interno di abilitazione interrupt (INTE); tali segnali sono definiti nel seguente modo:

- $\overline{OBF}$ *(Buffer d'Uscita Pieno).* Questo è il *segnale di strobe uscita dati* ed è generato dal PPI. Uno 0 logico su questa linea indica che il microcomputer ha scritto i dati nella porta e che questi si trovano ora nel latch.
- $\overline{ACK}$ *(Ingresso d'Accettazione).* Uno 0 logico su questa linea d'ingresso, indica che la periferica ha accettato dati dalle linee d'uscita delle porte. Esso è un segnale di accettazione proveniente dalla periferica.
- *INTR (Richiesta Interrupt).* Questa linea d'uscita viene usata per interrompere il microcomputer dopo che i dati sono stati inviati fuori dal PPI ed accettati dalla periferica. INTR, è posto al valore logico 1 quando $\overline{ACK}$, $\overline{OBF}$ e INTE sono tutti al valore logico 1, per indicare che i dati sono stati ricevuti dalla periferica.
- *INTE (Abilitazione Interrupt).* Questo è il flip-flop di abilitazione interrupt, come descritto per l'ingresso in modo 1. Come per le operazioni d'ingresso i flip-flop $INTE_A$ e $INTE_B$, possono ancora essere usati per abilitare o disabilitare, selettivamente, rispettivamente le porte A e B, questa volta per l'uscita in modo 1. Per le operazioni d'uscita del PPI, $INTE_A$ è controllato tramite il set e il reset di PC2. Si noti di nuovo che le operazioni di set/reset bit su

PC6 e PC2, per l'uscita in modo 1, sono operazioni su *flip-flop interni del PPI*. Gli stati logici dei pin PC6 e PC2 del PPI, non sono influenzati. nell'uscita in modo 1 questi pin del PPI sono usati per l'ingresso di $\overline{ACK}$ dalle periferiche esterne.

La Fig. 6-6, riporta un diagramma di temporizzazione che illustra la sequenza di funzionamento per un'operazione d'uscita in modo 1. L'operazione di scrittura iniziale, alla partenza del programma, avverrà probabilmente da parte della CPU, mentre la linea INTR è al valore logico 0. Dopo ciò, sarà stabilito il funzionamento a comando di interrupt, come illustrato nel diagramma di temporizzazione di Fig.6-6. Sup-

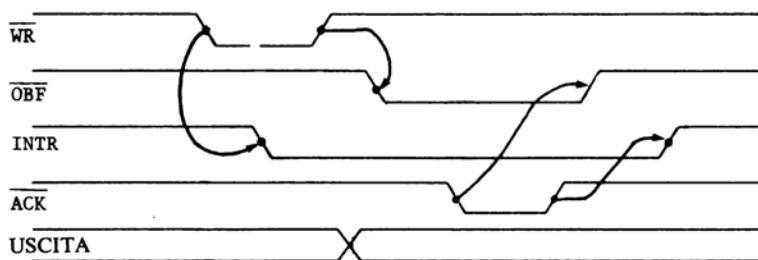



Fig. 6-6. Diagramma di temporizzazione per le operazioni d'uscita in modo 1. Si suppone che il flip-flop di abilitazione interrupt (INTE) sia stato posto al valore logico 1.

poniamo che, sia alla porta A che alla porta B, si sia stabilito il funzionamento d'uscita a comando di interrupt in modo 1. Ciò significa dire che si è accettata un'operazione di scrittura verso una periferica iniziata precedentemente, ponendo INTR al valore logico 1. Questo è il momento in cui possiamo supporre inizi il diagramma di temporizzazione di Fig. 6-6. Tramite l'impiego di un interrupt vettorizzato, o per mezzo di polling, la CPU deve poi individuare la porta che ha interrotto e iniziare una nuova sequenza di scrittura verso la porta d'uscita del PPI in modo 1. Il fronte iniziale di WR (Fig. 6-6), porta INTR al valore logico 0, rimuovendo l'interruzione della CPU.* La salita di WR, causa la discesa di $\overline{OBF}$ che, a turno, funge da strobe ai dati, sulle linee d'uscita della porta nei latch delle periferiche. Quando la periferica ha accettato i dati, pone al livello basso la propria linea $\overline{ACK}$, e ciò porta $\overline{OBF}$ al valore logico 1. Quando $\overline{ACK}$ va ad 1, la linea di richiesta interrupt, INTR, va ad 1, la CPU viene così interrotta ed inizia un altro ciclo. In conclusione, una sequenza di scrittura in modo 1, inizia con un interrupt. L'operazione di scrittura della CPU, causa la generazione di un impulso di strobe

* Questo *non* deve essere confuso con un'accettazione di interrupt.

Fig. 6-7. Combinazione ingresso/uscita delle porte A e B per il funzionamento del PPI in modo 1.

dati ($\overline{OBF}$) da parte del PPI, per lo strobe dati alla periferica. La periferica accetta la ricezione di questi dati portando basso $\overline{ACK}$. In seguito all'impulso $\overline{ACK}$ si genera un altro interrupt, per iniziare una nuova operazione di scrittura.

(C) Combinazione ingresso/uscita in Modo 1

Poichè le porte A e B possono essere programmate indipendentemente come ingresso o uscita, sono possibili due combinazioni ingresso/uscita in modo 1, come indicato in Fig. 6-7. Il nostro primo esempio di impiego del funzionamento in modo 1 del PPI per ingresso e uscita dati combinato (Fig. 6-1), può ora essere confrontato con la Fig. 6-7A, per vedere il modo in cui sono stati usati i bit di controllo della porta C. Si noti che le due linee della porta C del PPI che non sono usate per l'handshaking (PC6 e PC7), devono essere usate per comandare la copertura (blanking) e cancellazione della parola dei flag. Tali linee saranno comandate tramite la possibilità di set/reset bit.

6-3. CONDIZIONI DI FUNZIONAMENTO IN MODO 1

(A) Hardware

Nella maggior parte delle applicazioni di microcomputer, le periferiche vengono selezionate sulla base del costo e delle necessità globali del sistema. Bisogna poi aggiungere i compiti di interfacciamento di queste periferiche verso la CPU ed è qui che il PPI è più apprezzabile a causa della propria flessibilità. Se la periferica deve essere usata in un'applicazione di microcomputer, per fornire un'interfaccia di strobe dati/ accettazione del tipo uguale a quello del funzionamento in modo 1 del PPI, la *prima decisione hardware* che deve essere presa, nell'implementazione dell'interfaccia in Modo 1, è se si devono usare le porte A e B per l'ingresso, l'uscita o una combinazione delle due. Una volta che questa decisione è stata presa, i segnali di handshaking della porta C sono automaticamente definiti (cf Fig. 6-2) e si possono collegare alla periferica gli opportuni bit di strobe ed accettazione dati.
La *seconda decisione hardware* da prendere, è il tipo di struttura di interrupt che si vuole usare, cioè interrupt vettorizzato e interrupt in polling. L'*interrupt vettorizzato* in genere, permette una risposta più veloce della CPU alla periferica interrompente, dal momento che la CPU viene automaticamente portata ad eseguire l'apposita routine di servizio per mezzo di una delle istruzioni, ad un solo byte, di chiamata o di restart, RST, dell'8080. Un ulteriore three-state, buffer latch quale ad

Fig. 6-8. Schema del circuito di un PPI con interrupt vettorizzato in modo 1 per l'interfaccia a un microcomputer basato sull'8080.

esempio l'SN74365, è necessario per "spingere" l'istruzione RST nel registro istruzione dell'8080A, quando la CPU accetta l'interrupt con un valore logico basso sull'uscita dell'accettazione interrupt ($\overline{INTA}$ o $\overline{IACK}$). La Fig. 6-8, illustra un circuito che può essere usato per implementare interrupt vettorizzati, con un PPI avente un dispositivo d'ingresso alla porta A e un dispositivo d'uscita alla porta B. Si noti che si sono usati i segnali IBF e $\overline{OBF}$ come ingressi dell'SN74365. È utile notare qui, che il circuito non supporterà interrupt simultanei e che la priorità nel servizio degli interrupt deve essere decisa nel software.

Ora con riferimento ai diagrammi di temporizzazione, rispettivamente per l'ingresso e l'uscita in modo 1, di Fig. 6-4 e 6-6, si nota che, sia IBF che $\overline{OBF}$, sono ad un valore logico alto nell'intervallo di tempo in cui sono alti $INTR_A$ e $INTR_B$. Da questo momento, in questo esempio, si useranno, rispettivamente per l'ingresso e l'uscita, le istruzioni di RESTART di interrupt vettorizzato RST 7 (377) e RST 5 (357). Si noti anche che, avendo l'8080A solamente una linea di interrupt (INT), i

Fig. 6-9. Parola di stato del modo 1, ottenuta leggendo la porta C.

segnali di $INTR_A$ e $INTR_B$ sono stati messi in OR e l'uscita risultante è stata collegata a INT.

Se è richiesta solamente una porta per l'I/O in modo 1, si può effettivamente usare una caratteristica del chip 8228, controllore di sistema, per spingere durante un interrupt un'istruzione di vettore di interrupt RST 7 nel registro d'istruzione dell'8080A. Tutto ciò che è richiesto, è il collegamento della linea $\overline{INTA}$ dell'8228 al +12V attraverso una resistenza di 1 KΩ. Ciò evita poi la necessità del buffer three-state SN74365 indicato in Fig. 6-8. In un sistema a microcomputer che non usi un 8228, un codice RST 7 (377), potrà essere ottenuto per mezzo di un SN74365 come in Fig. 6-8, ma con tutte le linee d'ingresso collegate ad un 1 logico.

Se in un'interfaccia a comando di interrupt si impiegano due o più porte in modo 1, si può ancora usare una sola istruzione RST, se si controllano poi le porte in modo 1 per determinare quale stia richiedendo servizio. Questo è l'approccio di *interrupt in polling*. La domanda che nasce poi è come determinare lo stato di ogni porta in modo 1, dal momento che tale fatto non era considerato in precedenza. La risposta è quella di iniziare un'operazione di lettura dalla porta C. Quando si è fatto ciò e l'8255 è programmato per il funzionamento in modo 1, il microcomputer riceverà la *parola di stato del modo 1*, che è riportata in Fig. 6-9. Lo stato delle linee INTR (bit D3 e D0), può poi essere controllato dal software, dal momento che i bit della parola di stato rappresentano, in generale, lo stato delle linee della porta C associate.

Confrontando la parola di stato del modo 1 di Fig. 6-9, con le assegnazioni dei pin della porta C per l'ingresso e l'uscita, riportate in Fig. 6-2, si vedrà che le uniche differenze sono le sostituzioni in Fig. 6-9 di:

(a) Le linee $\overline{STB}$ (PC4 e PC2) con il flip-flop di stato INTE per l'ingresso.

(b) Le linee $\overline{ACK}$ (PC6 e PC2) con il flip-flop di stato INTE per l'uscita.

L'approccio dell'interrupt in polling, richiede più software per determinare quale sia la porta che ha generato interrupt. Esso ha quindi un tempo di risposta al servizio maggiore, che comunque è ancora dell'ordine di sole poche decine di microsecondi.

(B) Software

Il primo passo nella scrittura del software per il funzionamento in modo 1 del PPI, è quello di scrivere i codici per l'inizializzazione del PPI per configurarlo come stabilito dalle considerazioni sull'hardware. Ciò si ottiene caricando nell'accumulatore l'opportuno byte di controllo del modo e inviandolo in uscita al registro di controllo del PPI. Il byte di controllo del modo è determinato nel solito modo, facendo riferimento al formato della parola di controllo del modo di Fig. 2-2A. Notare che in questo caso i bit D6, D5 e D2, sono posti rispettivamente a 0,1 ed 1 al fine di programmare le porte A e B in modo 1.

La seconda operazione nell'inizializzazione del PPI in modo 1, è quella di predisporre i flip-flop di abilitazione interrupt, INTE, tramite la possibilità di set/reset bit della porta C. Attenzione però ad assicurarsi che si mettano a uno i bit giusti, dal momento che *i bit della porta C assegnati per i flag INTE delle porte A e B, cambiano per l'ingresso e l'uscita* (cf Fig. 6-9). I bit PC4 e PC2 della porta C, sono assegnati ai flip-flop INTE per l'uscita in modo 1.

Il rimanente software associato al funzionamento ingresso/uscita in modo 1 del PPI, è relativo al servizio interrupt ($INTR_A$ e $INTR_B$) generato dalle porte A e B. I requisiti software per un handshaking di I/O con interrupt in polling e con interrupt vettorizzato, sono illustrati in Fig. 6-10 e possono essere confrontati. In genere sono richiesti più step per il servizio di un handshaking di I/O con interrupt in polling, dal momento che ad ogni interrupt, si deve leggere la parola di stato del modo 1 ed occorre controllare separatamente lo stato delle linee INTR della porta A e di quella B (cf bit D0 e D3 di Fig. 6-9), per determinare quale porta stia richiedendo servizio. La tecnica di programmazione usata per determinare lo stato di una porta, è lo stesso che si è usato nel Capitolo 5 per determinare lo stato di una periferica (cf Paragrafo 5-3B), cioè l'impiego di una maschera combinato con un salto di programma basato sul

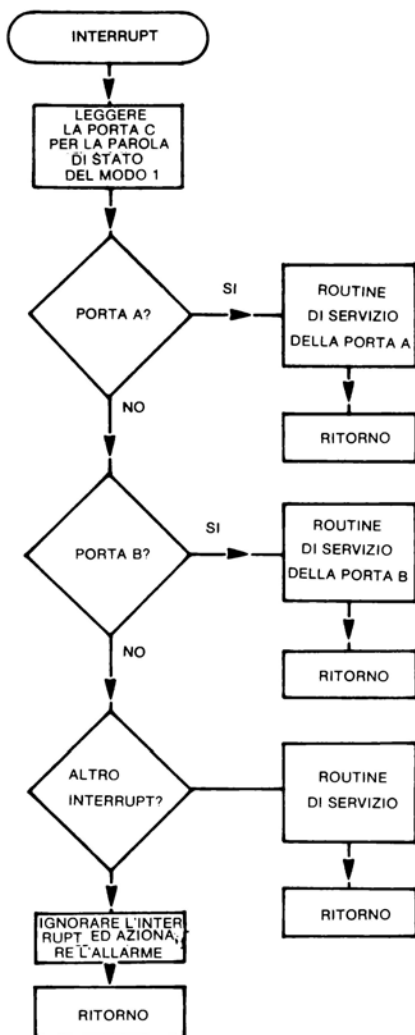
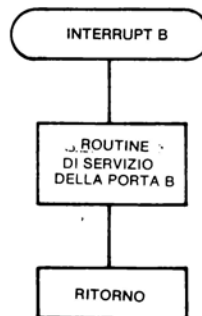



Fig. 6-10. Diagrammi di flusso delle subroutine di interrupt necessarie per fornire interrupt in polling e vettorizzato nel funzionamento del PPI in modo 1.

risultato di un AND logico fra il byte di stato e quello della maschera. Tale tecnica è illustrata nell'esempio descritto nel Paragrafo 6-4. La velocità di risposta di una subroutine di handshaking di I/O con inter-

rupt in polling, può essere aumentata, in quelle circostanze in cui si sa che una periferica richiederà servizio più frequentemente di altre. In questi casi è chiaramente più efficiente controllare in primo luogo lo stato della periferica che si ritiene interrotta più spesso. La porta il cui stato è controllato per primo dal programma, si dice che ha la più alta *priorità*. Quando si scrive il software per interrupt in polling, la priorità delle periferiche del sistema deve essere decisa in modo che lo stato di ogni porta possa essere controllato nell'ordine più adeguato. Tali decisioni possono essere prese in base alla velocità di risposta, o forse sulla base di qualche altro criterio, quale l'importanza relativa dei dati provenienti da una o da tutte le periferiche.

Il software per interrupt vettorizzati, come mostrato in Fig. 6-10B, è molto più semplice di quello per interrupt in polling e richiede solamente la scrittura delle routine di servizio di ogni porta, alla giusta locazione del vettore di restart. A causa di questa semplicità di programmazione, il tempo di risposta per il servizio della periferica, è minore di quello per l'handshaking di I/O con interrupt in polling. Con interrupt vettorizzati, è ancora utile considerare la priorità di servizio delle periferiche. In questo caso, comunque, la priorità di servizio sarà in generale, impostata in hardware anzichè in software, ciò è trattato nel Capitolo 23 del *TTL e 8080A - VOL. 2: Elettronica Digitale, Tecniche di Programmazione e Interfacciamento dei Microcomputer* (Gruppo Editoriale Jackson). In quella sede viene anche discusso un circuito che stabilisce la priorità del servizio delle periferiche in hardware, mediante un codificatore di priorità SN74148.

Nei sistemi a microcomputer in cui le esigenze di elaborazione dati e di sequenze logiche della CPU sono minime, può essere accettabile il *polling continuo* delle linee di stato INTR del PPI. Diagrammi di flusso, del software necessario per fornire un polling continuo per operazioni di lettura e scrittura I/O, sono riportati in Fig. 6-11. Un piccolo vantaggio di questo approccio, è che non è necessario nessun hardware esterno oltre al PPI.

6-4. UN ESEMPIO

Per illustrare le considerazioni conseguenti all'uso del funzionamento in modo 1 del PPI, consideriamo ora l'esempio di Fig. 6-1. In esso una tastiera completamente decodificata ed un video display, sono stati interfacciati al PPI. Esamineremo le considerazioni hardware e poi quelle software.

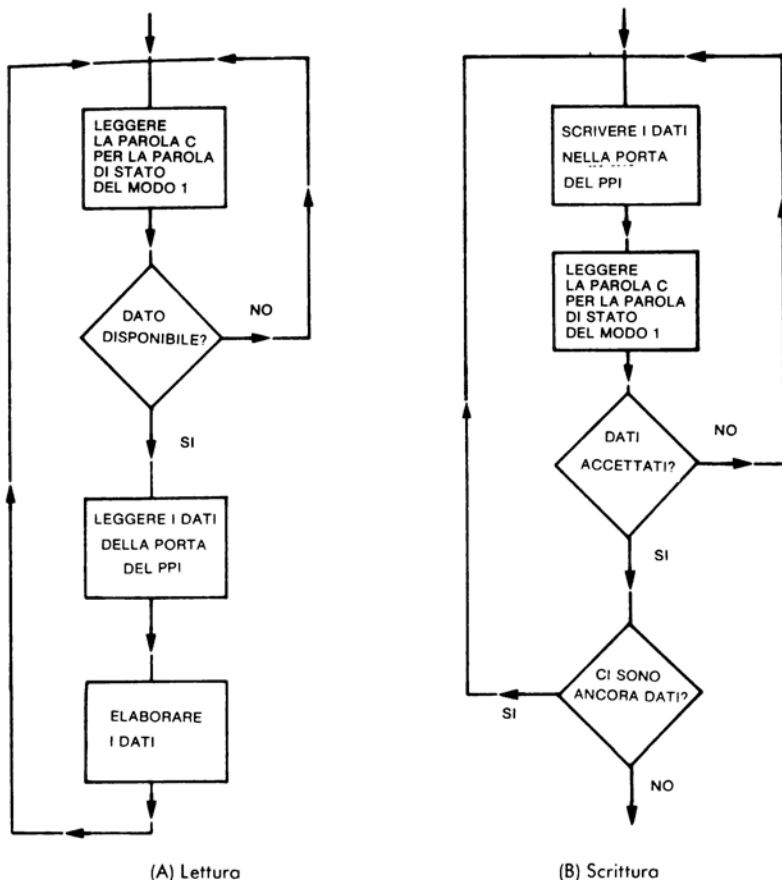



Fig. 6-11. Diagrammi di flusso per polling software continuo nel funzionamento del PPI in modo 1.

(A) Hardware

Nel produrre lo schema di Fig. 6-1, sono già state fatte parecchie scelte importanti. Chiaramente, entrambe le periferiche forniscono un'interfaccia strobe dati/accettazione e si è così deciso di usare il funzionamento in modo 1 del PPI. Le restanti linee I/O della porta C, sono state poi assegnate al display video come linee di controllo. Queste decisioni sono state prese per adattare il PPI alle esigenze della periferica. Avendo deciso di impiegare un sistema a comando di interrupt anzichè un sistema a continuo polling software, il problema successivo è quello di

decidere fra gli approcci ad interrupt vettorizzato e ad interrupt in polling. La decisione verrà presa solitamente, sulla base delle esigenze di velocità di risposta e sul compromesso fra hardware aggiuntivo e software extra. L'autore consiglia, se non è richiesta la massima velocità di risposta e se non interessa il risparmio di memoria, di impiegare l'approccio ad interrupt in polling, dal momento che esso riduce l'hardware esterno a una porta d'istruzione interrupt e ad un solo gate OR. Questo è l'approccio che sarà usato.

Il circuito d'interfaccia richiesto, è riportato in Fig. 6-8 con l'SN74365 collegato in hardware per avere il vettore 367 (RST 6). Poichè si usa un approccio a interrupt in polling, la priorità nel servizio delle periferiche sarà determinata dal software invece che dall'hardware. Assegneremo la massima priorità al display, dal momento che esso richiederà il refresh (rinfresco) più spesso dell'ingresso dati nella tastiera. Infine la decisione d'impiegare la porta A come ingresso e la B come uscita, è abbastanza arbitraria.

(B) Software

Il primo passo, quando si scrive il software, è quello di disegnare un diagramma di flusso. La Fig. 6-10A, mostra il diagramma di flusso per il software dell'interrupt in polling. Non si farà il flowchart per le routine di servizio delle porte A e B, dal momento che queste sono funzione del particolare sistema a microcomputer usato per la tastiera e il display. Prima di scrivere i codici del flowchart di interrupt in polling comunque, si scriveranno i codici per l'inizializzazione del PPI. Ciò è riportato nel listing del programma di Fig. 6-12 insieme con la routine di servizio dell'interrupt in polling. Nella scrittura del programma l'autore, ha cominciato con i codici d'inizializzazione del PPI e con la routine di servizio d'interrupt in polling, sotto forma di mnemonici, aggiungendo dei commenti per descrivere la struttura del programma. A variabili come il byte di controllo del modo, byte di set bit, byte di maschera e indirizzi del PPI, sono stati attribuiti nomi simbolici. Si legga ora il listing del programma controllando quei punti.

Programma 6-1 (Fig. 6-12)

Una caratteristica importante di questo programma, è la determinazione dei valori delle variabili. Gli indirizzi del registro di controllo e della porta C del PPI, sono funzione del modo in cui il PPI è collegato

```
                *000 060
000 060 303              JMP        /SALTARE AL VETTORE DI RESTART DELLA
000 061 060              RESTRT     /SUBROUTINE
000 062 003              0
                /
                /
                /
                /
                /
                /IL PROGRAMMA INIZIA QUI
                /
                DB MODE 264
                DB PORTC 202
                DB CNTRL 203
                DB INTEAS 011
                DB INTEBS 005
                DB MASKA 010
                DB MASKB 001
                DW PASVC 003 300
                DW PBSVC 003 200
                *003 001
003 001 076     START, MVIA         /INIZIALIZZARE LA PORTA A DEL PPI: MODO 1
003 002 264              MODE       /INGRESSO; PORTA B: MODO 1 USCITA
003 003 323              OUT        /INVIARE IL BYTE DI CONTROLLO DEL
                                    /MODO AL
003 004 203              CNTRL      /REGISTRO DI CONTROLLO DEL PPI
003 005 076              MVIA       /SET DI INTEA (PC4) USANDO IL BYTE DI
003 006 011              INTEAS     /CONTROLLO SET/RESET BIT
003 007 323              OUT
003 010 203              CNTRL
003 011 076              MVIA       /SET DI INTEB (PC2) USANDO IL BYTE DI
003 012 005              INTEBS     /CONTROLLO SET/RESET BIT
003 013 323              OUT
003 014 203              CNTRL
003 015 373              EI
                /
                /QUESTO È L'INIZIO DELLA
                /ROUTINE DI SERVIZIO INTERRUPT
                /
                *003 060
003 060 365     RESTRT,PUSHPSW      /SALVARE LO STATO DEL MICROCOMPUTER
003 061 345              PUSHH
003 062 325              PUSHD
003 063 305              PUSHB
003 064 333              IN         /INSERIRE LA PAROLA DI STATO DEL
                                    /MODO DALLA
003 065 202              PORTC      /PORTA C
003 066 107              MOVBA      /SALVARE LA PAROLA DI STATO DEL MODO 1
003 067 346              ANI        /È LA PORTA B CHE HA INTERROTTO?
003 070 001              MASKB      /BYTE MASCHERA PER INTRB (D0)
003 071 302              JNZ        /SI'! - SALTARE AL SERVIZIO DELLA
003 072 200              PBSVC      /PERIFERICA PORTA B
003 073 003              0
003 074 170              MOVAB      /NO - RIPRISTINARE LA PAROLA DI STATO
003 075 346              ANI        /È LA PORTA A CHE HA INTERROTTO?
003 076 010              MASKA      /BYTE MASCHERA PER INTRA (D3)
003 077 302              JNZ        /SI'! - SALTARE AL SERVIZIO DELLA
003 100 300              PASVC      /PERIFERICA PORTA A
003 101 003              0
003 102 301              POPB       /NO, IGNORARE L'INTERRUPT
003 103 321              POPD       /RIPRISTINARE LO STATO DEL
                                    /MICROCOMPUTER
003 104 341              POPH
003 105 361              POPPSW
003 106 311              RET        /E RITORNARE
```

Fig. 6-12. Programma 6-1.

alle linee indirizzi del microcomputer e così in questo esempio, si sono usati rispettivamente 203 e 202. Le rimanenti variabili sono state determinate come segue:

(a) *MODO*. Questo è il byte di controllo del modo, che è richiesto per programmare la porta A per il funzionamento d'ingresso in modo 1, la porta B per il funzionamento d'uscita in modo 1 ed i bit superiori della porta C, PC6 e PC7, come uscite. Dalla Fig. 2-2A, si ha che il byte richiesto è 10110100 oppure 264.

(b) *INTEAS, INTEBS*. Questi sono i byte di controllo set/reset bit, necessari per impostare rispettivamente $INTE_A$ e $INTE_B$, flag di abilitazione interrupt, delle porte A e B. Con riferimento alla parola di stato del modo 1 di Fig. 6-9, si vede che il flag di abilitazione interrupt della porta A ($INTE_A$), è rappresentato da bit diversi della porta C, in funzione del fatto che la porta A può essere usata come ingresso o come uscita. In questo esempio la porta A è usata come ingresso e quindi occorre mettere al valore logico 1 il bit D4 della porta C, usando la possibilità di set/reset bit della porta C del PPI. Facendo riferimento alla Fig. 2-2B, il byte di controllo set/reset bit richiesto è 00001001, oppure 011. Il flip-flop di abilitazione interrupt della porta B, è controllato impostando il bit D2 della porta C (cf Fig. 6-9). Il byte di controllo set/reset bit richiesto (da Fig. 2-2B) per mettere ad uno PC2 è 00000101 = 005.

(c) *MASKA, MASKB*. Questi sono i byte di maschera necessari per eliminare tutti i bit della parola di stato nel modo 1, eccetto quelli che danno lo stato dei flag di richiesta di interrupt della porta A e della B, cioè rispettivamente $INTR_A$ e $INTR_B$. Per la porta A come ingresso, $INTR_A$ è rappresentato dal bit D3 della parola di stato del modo 1 (Fig. 6-9) e quindi MASKA è 00001000 oppure 010. $INTR_B$ è rappresentato dal bit D0 della parola di stato del modo 1 e così si ha 00000001 oppure 001.

(d) *PASVC e PBSVC*. Questi rappresentano i byte meno significativi degli indirizzi delle routine di servizio della porta A e della B rispettivamente e in questo programma sono stati arbitrariamente posti rispettivamente a 300 e 200.

In conclusione, questo esempio è stato presentato per illustrare l'approccio di progetto per l'impiego del PPI per lo handshaking di I/O in modo 1. Per l'hardware i punti principali da esaminare sono:

- Assegnazione della porta e della linea di controllo.
- Interrupt vettorizzato, interrupt in polling o polling da software.
- Priorità della periferica.

- Se si ha interrupt vettorizzato o interrupt in polling, occorre poi forzare le istruzioni RST sul bus dati durante in interrupt.

Per l'hardware le principali considerazioni sono:

- Scrittura dei codici d'inizializzazione.
- Per l'interrupt vettorizzato, scrittura o opportuna allocazione in memoria delle routine di servizio periferiche.
- Per interrupt in polling, scrittura ed opportuna allocazione in memoria del software per determinare quale periferica ha interrotto.

Le variabili che devono essere necessariamente determinate sono:

- Il byte di controllo del modo.
- I byte di controllo set/reset bit, necessari per impostare $INTE_A$ e/o $INTE_B$.
- Il byte di maschera, necessario per determinare lo stato di $INTR_A$ e/o $INTR_B$.
- Gli indirizzi del PPI per le porte da A a C e per il registro di controllo.

6-5. SOMMARIO DEGLI ESPERIMENTI DA 6-1 A 6-5

Esperimento	*Descrizione*
6-1	Lo scopo di questo esperimento, è quello di mostrare il funzionamento *d'uscita in modo 1* dell'8255, impiegando la tecnica del *polling continuo da software*. Viene anche illustrato l'uso dell'handshaking fra il microcomputer ed il generatore d'impulsi.
6-2	Lo scopo di questo esperimento è quello di mostrare il funzionamento *d'ingresso in modo 1* dell'8255, impiegando la tecnica del *polling continuo da software.*
6-3	Questo esperimento descrive il *funzionamento in modo 1* dell'8255 per *ingresso e uscita combinati* con *polling continuo da software.*
6-4	Questo esperimento illustra il funzionamento con *interrupt il polling* in modo 1 dell'8255.
6-5	Lo scopo di questo esperimento è quello di descrivere il funzionamento con *interrupt vettorizzato* in modo 1 del PPI.

ESPERIMENTO 6-1
FUNZIONAMENTO D'USCITA IN MODO 1 DEL PPI

Scopo

Lo scopo di questo esperimento è quello di descrivere il *funzionamento d'uscita in modo 1* del chip circuito integrato 8255. Sarà illustrato anche l'impiego di handshaking fra il microcomputer ed il generatore d'impulsi.

Passo 1

La Fig. 6-15, mostra le indicazioni della Intel Corporation sul funzionamento in modo 1 delle porte A e B come porte d'uscita.

In questo esperimento si dovrà impiegare la parola di controllo del modo 240, la quale programma la porta A come una porta d'uscita in modo 1. Il bit D3 della parola di controllo, è al valore logico 0, il che significa che PC4 e PC5 sono stati programmati come uscite.

Passo 2

Collegare il circuito riportato nello schema (Fig. 6-14). Collegare PC6 ($\overline{ACK_A}$) all'uscita "1" del generatore d'impulsi. Questa è all'uscita del generatore d'impulsi che è normalmente al valore logico 1.

Passo 3

Caricare il programma in memoria. Spiegare cosa significano le parole di controllo ai L0 indirizzi di memoria 001 e 002.
A quali indirizzi del programma esse vengono caricate nel registro di controllo del chip 8255?

Schema del circuito (Fig. 6-13)

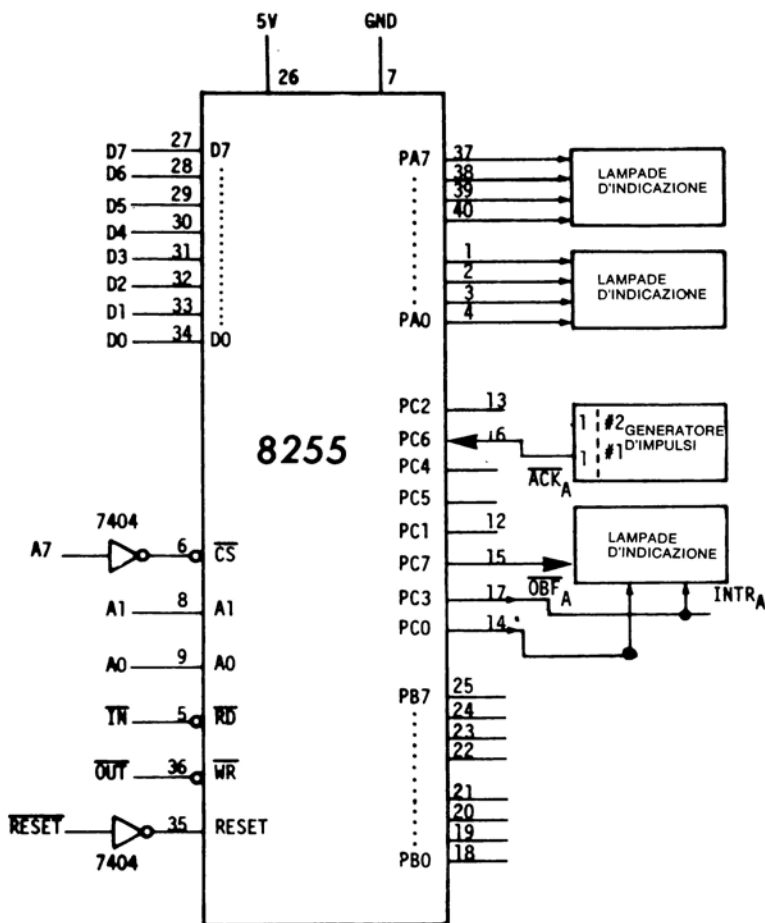



Fig. 6-13. Circuito per l'Esperimento 6-1.

Il byte di controllo al L0 indirizzo di memoria 001, è quello del modo ed è caricato nel registro di controllo del PPI con le istruzioni che iniziano al L0 indirizzo di memoria 003. Il byte di controllo al L0 indirizzo 002, è quello di set/reset bit, che è inviato al registro di controllo con le istruzioni che iniziano al L0 indirizzo di memoria 010.

Programma (Fig. 6-14)

```
               /
               /OPERAZIONI D'USCITA NEL PPI IN MODO 1
               /
               DB CNTRL 203
               DB DATA1 200
               DB DATA3 202
               *003 000
003 000 001          LXIB     /CARICARE NELLA COPPIA DI REGISTRI B I
                              /SEGUENTI BYTE
003 001 240          240      /PAROLA DI CONTROLLO DEL MODO DEL PPI
003 002 015          015      /PAROLA DI CONTROLLO DI SET/RESET BIT
                              /DEL PPI
003 003 171          MOVAC    /METTERE LA PAROLA DI CONTROLLO DEL
                              /MODO NELL'ACCUMULATORE
003 004 323          OUT      /INVIARE IL CONTENUTO
                              /DELL'ACCUMULATORE AL
003 005 203          CNTRL    /REGISTRO DI CONTROLLO PER PPI
003 006 026          MVID     /CARICARE 377 NEL REGISTRO D
003 007 377          377
003 010 170          MOVAB    /CARICARE IN A IL CONTROLLO SET/RESET
                              /BIT DEL PPI
003 011 323          OUT      /INVIARLO AL
003 012 203          CNTRL    /REGISTRO DI CONTROLLO DEL PPI
003 013 172    LOOP, MOVAD    /AZZERARE L'ACCUMULATORE
003 014 323          OUT      /INVIARE IL CONTENUTO
                              /DELL'ACCUMULATORE ALLA
003 015 200          DATA1    /PORTA A
003 016 024          INRD     /INCREMENTARE DI UNO IL CONTENUTO
                              /DEL REGISTRO D
003 017 172          MOVAD
003 020 323          OUT      /VISUALIZZARE IL PROSSIMO BYTE
                              /D'USCITA
003 021 001          001      /ALLA PORTA = 1
003 022 333    WAIT, IN       /INSERIRE LO STATO DELL'8255, CIOE'
003 023 202          DATA3    /IL CONTENUTO DELLA PORTA C
003 024 323          OUT      /VISUALIZZARE LO STATO DEL PPI ALLA
003 025 000          000      /PORTA = 0
003 026 346          ANI      /AND DEL SEGUENTE BYTE DI
                              /MASCHERA CON
003 027 010          010      /IL CONTENUTO DELL'ACCUMULATORE.
                              /CIO' MASCHERA
                              /TUTTI I BIT ECCETTO IL BIT 3 (INTRA)
003 030 312          JZ       /INTRA = 1; NO : - PROVA DI NUOVO
003 031 022          WAIT
003 032 003          0
003 033 303          JMP      /SI', BUFFER D'USCITA DEL PPI VUOTO
                              /QUINDI SALTA A
003 034 013          LOOP     /INVIARE IN USCITA UN ALTRO BYTE
003 035 003          0
```

Fig. 6-14. Programma per l'Esperimento 6-1.

Passo 4

Eseguire il programma. Avviare e bloccare il generatore d'impulsi diverse volte. Che cosa si osserva alla porta d'uscita A?

Abbiamo visto che l'uscita alla porta A, era incrementata di uno ogni volta che il generatore d'impulsi veniva avviato e fermato. Mentre esso era avviato (mandando PC6 ($\overline{ACK_A}$) al valore logico 0), PC7 ($\overline{OBF_A}$) andava al valore logico 0. L'uscita PC3 ($INTR_A$) rimaneva al livello logico 0, in accordo con l'indicazione visiva. Ad ogni modo, quando si è impiegato il semplice circuito di conteggio riportato in Fig. 6-16, si è osservato un incremento ogni volta che il generatore veniva avviato e

Definizione del Segnale di Controllo d'Uscita

$\overline{OBF}$ (Buffer d'Uscita Pieno F/F)

L'uscita $\overline{OBF}$ andrà "bassa" per indicare che la CPU ha scritto dei dati nella porta specificata. L'OBF FBF sarà posto ad uno dal fronte di salita dell'ingresso WR e sarà azzerato dal fronte di discesa del segnale d'ingresso $\overline{ACK}$.

$\overline{ACK}$ (Ingresso Accettazione)

Un valore "basso" su questo ingresso, informa l'8255 che i dati provenienti dalla porta A, o dalla B, sono stati accettati. In assenza, una risposta dal dispositivo periferico, indica che esso ha ricevuto i dati inviati in uscita dalla CPU.

INTR (Richiesta Interrupt)

Un valore "alto" su questa uscita può essere usato per interrompere la CPU, quando un dispositivo ha accettato i dati trasmessi dalla CPU stessa. INTR è posto ad uno dal fronte di salita di $\overline{ACK}$, se $\overline{OBF}$ e INTE sono a "uno". Esso è azzerato dal fronte di discesa di $\overline{WR}$.

INTE A

Controllato dal set/reset bti di PC6

INTE B

Controllato dal set/reset bid di PC2

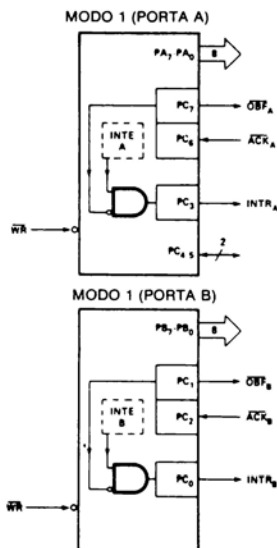



Fig. 6-15. Funzionamento delle porte A e B come porte di uscita in modo 1.

bloccato. Ciò significa che il collegamento PC3, ha prodotto un impulso ogni volta che il generatore d'impulsi è stato avviato e bloccato.

Passo 5

Esaminare attentamente i passi del programma; si è in grado di individuare quelli che costringono il microcomputer in un "loop" una volta che si esegue il programma? Se si trovano difficoltà a rispondere a questa domanda, si esamini il diagramma delle temporizzazioni per l'uscita in modo 1, riportato in Fig. 6-17. Si consideri la relazione fra $\overline{ACK_A}$ (PC6) e $INTR_A$ (PC3).

Quando il programma viene fatto partire per la prima volta, il PPI è inizializzato per il funzionamento in modo 1 (L0 indirizzi da 000 fino a 012), con la porta A configurata come uscita. Il microcomputer trasferisce quindi i dati a tale porta (L0 indirizzi da 013 fino a 015). Ciò porta al

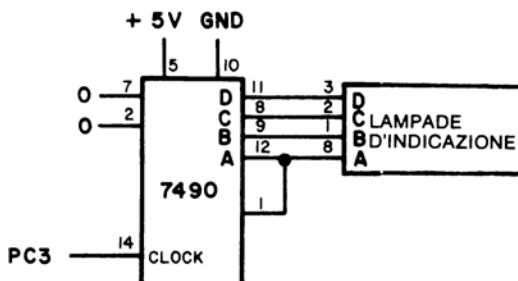



Fig. 6-16. Semplice circuito di conteggio.

valore logico 0 il flag di Buffer d'Uscita Pieno ($\overline{\text{OBF}}$), per indicare che i dati sono stati trasferiti alla porta A. Il programma a questo punto attende nel loop WAIT (L0 indirizzi da 022 fino a 032), fino a che non rileva la presenza del flag INTR_A, il quale è controllato in software. Il segnale INTR_A, viene posto al valore logico 1 quando si riconosce ($\overline{\text{ACK}_\text{A}}$) di avere ricevuto (la periferica) il dato. Una volta riconosciuta la ricezione del dato, si invia in uscita un nuovo (incrementato) byte di dati e si ripete il processo. Che cosa succede quando si avvia il generatore

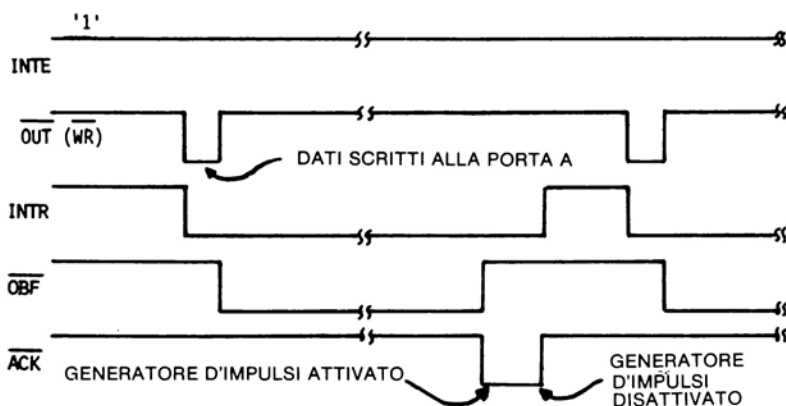



Fig. 6-17. Temporizzazione d'uscita in modo 1.

d'impulsi ($\overline{ACK_A} = 0$) e lo si mantiene in funzione? Per rispondere ci si serva dei diagrammi delle temporizzazioni di Fig. 6-17.

La risposta è che l'ingresso $\overline{ACK_A}$ del PPI, va al valore logico 0 e l'uscita $\overline{OBF_A}$ al valore logico 1. Per tutto il tempo che il generatore d'impulsi resta in funzione non accade nient'altro.

Passo 6

Si spieghi ora che cosa avviene quando si disattiva il generatore d'impulsi. (*Suggerimento*: Che cosa succede sul fronte di salita di $\overline{ACK_A}$ se sia $INTE_A$ che $\overline{OBF_A}$ sono al valore logico 1?). Spiegare che cosa succede all'uscita sulla porta A. Si prega di dare una risposta dettagliata.

Quando si disattiva il generatore d'impulsi $\overline{ACK_A}$ ($\overline{ACK_A} = 1$), si accetta la ricezione del dato che era stato precedentemente messo in uscita sulla porta A dall'8080 e visualizzato sulle lampade d'indicazione. In uscita viene poi inviato un nuovo byte di dati. Ciò è indicato dal flag di Buffer d'Uscita pieno ($\overline{OBF_A}$), che va nuovamente al valore logico 0.

Passo 7

Cambiare la parola di controllo set/reset bit, al L0 indirizzo 002 con 014. Quando questo byte viene inviato fuori al registro di controllo, esso causerà il reset al valore logico 0 del flip-flop interno del PPI, $INTE_A$. Ciò influenzerà il funzionamento dell'interfaccia?

Sì. "Esso disattiva" gli interrupt della porta A, cosicchè $INTR_A$ non può essere posto al valore logico 1.

Passo 8

Eseguire il programma con la variazione software (fatta nel passo 7). Avviare e fermare ripetutamente il generatore d'impulsi. Viene trasferito *qualche* dato alla porta A?

Sì, il dato è trasferito, ma solo il primo byte, 377.

Spiegare perchè la variazione del programma ha un tale effetto. Per la spiegazione può essere utile il diagramma di temporizzazioni di Fig. 6-18. Si noti che l'uscita, $INTR_A$, non va mai al valore logico 1. Si può confermare questo comportamento, con l'aiuto del contatore 7490 indicato nel passo 4. Con l'abilitazione interna di interrupt ($INTE_A$) disattivata, il segnale di accettazione non è spinto attraverso PC3, cioè l'uscita $INTR_A$.

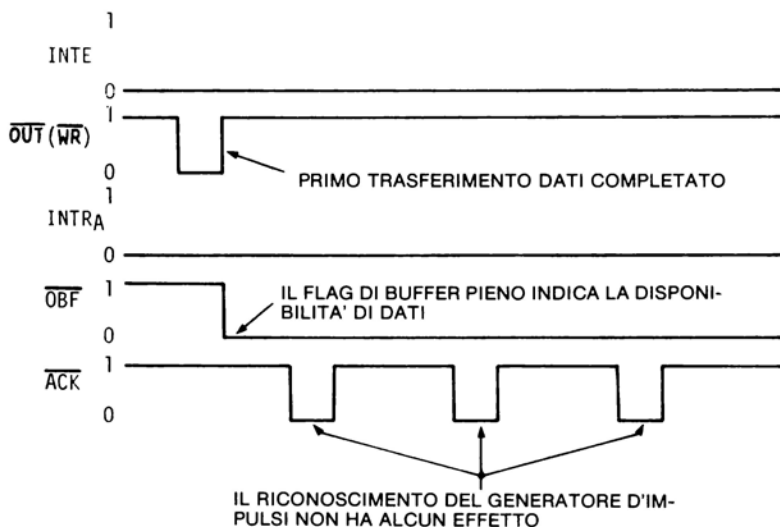



Fig. 6-18. Diagramma di temporizzazione per cambiamento del programma.

Domande

1. Un passo chiave del programma, è l'istruzione IN data al L0 indirizzo di memoria 022. Se si è risposto correttamente alle domande poste nei precedenti passi, si può già comprendere perchè lo stato del chip 8255 viene caricato nell'accumulatore.

2. La parola di stato per il funzionamento in modo 1, quando la porta A o la porta B sono porte d'uscita, è data dalla Intel, come in Fig. 6-19.

   (a) Qual'è il significato dei bit D3, D6 e D7?

   (b) Qual'è il significato del byte di maschera 010 al L0 indirizzo di memoria 027? Perchè si usa quello invece di 300 o 100?

Fig. 6-19. Parola di stato del funzionamento in modo 1 con la porta A, o la B, quale porta d'uscita.

3. Quali variazioni occorre fare al programma dato in questo esperimento, per permettergli di essere usato per il funzionamento d'uscita in modo 1 della porta B? *Suggerimento:* I cambiamenti devono essere fatti ai seguenti L0 indirizzi di memoria.

L0 Indirizzi di Memoria	*Uscita Porta A*	*Uscita Porta B*
001	240	
002	015	
015	200	
027	010	

4. Completare lo schema dato in Fig. 6-20 per il funzionamento d'uscita in modo 1 della porta B. Si vogliono otto lampade indicatrici alla porta B, un ingresso generatore d'impulsi su un adeguato ingresso della porta C e le uscite $\overline{OBF}_B$ e INTR, collegate ad un paio di lampade di indicazione.

ESPERIMENTO 6-2
FUNZIONAMENTO D'INGRESSO IN MODO 1 DEL PPI

Scopo

Lo scopo di questo esperimento è quello di mostrare il *funzionamento d'ingresso in modo 1* del chip circuito integrato 8255. Viene anche illustrato l'uso dell'handshaking.

Schema del circuito (Fig. 6-21)

NOTA: Se si è appena completato l'Esperimento 6-1, si avrà altro hardware collegato all'interfaccia. Esso non è riportato in questo schema al fine di maggior chiarezza.

Passo 1

L'autore consiglia di usare il contatore per dieci 7490 di Fig. 6-23, per seguire e visualizzare il comportamento del programma di Fig. 6-22 e del circuito 8255. Collegare l'uscita PC0 ($INTR_B$) al contatore di decade 7490.

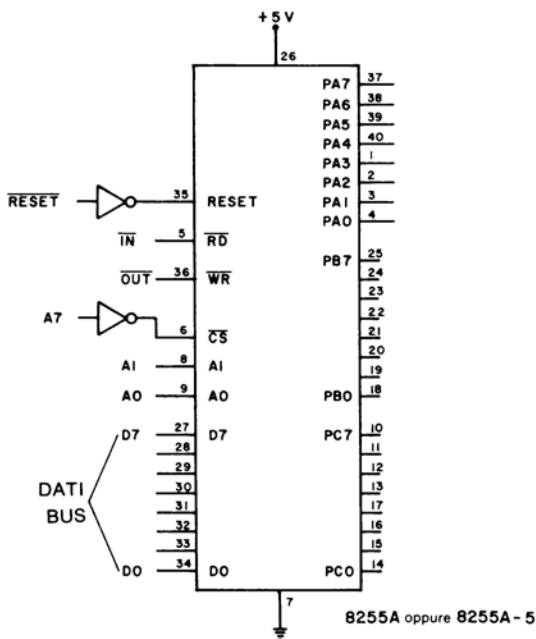



Fig. 6-20. Schema da completare.

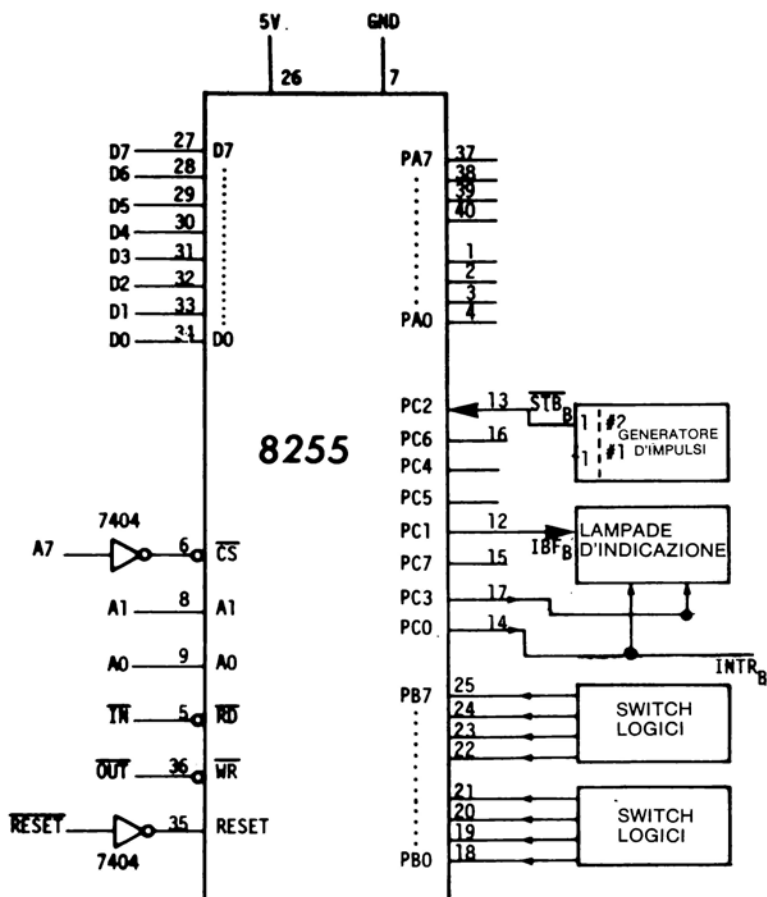



Fig. 6-21. Circuito per l'Esperimento 6-2.

Passo 2

In Fig. 6-24, sono riportate le informazioni della Intel Corporation sul funzionamento in modo 1 delle porte A e B come *ingressi.*

In questo esperimento si userà la parola di controllo del modo 246, per programmare la porta B come ingresso. Il bit D3, è al valore logico 0, il che significa che PC6 e PC7 sono programmati per il funzionamento d'uscita. La parola di stato della porta C per il funzionamento in modo 1, quando la porta A o B sono porte d'ingresso, è data dalla Intel come da Fig. 6-25. Il bit D0 ($INTR_B$) di questa parola di stato della porta C è

posto al valore logico 1, quando il dato è disponibile alla porta B, per l'ingresso nel microcomputer (supponendo che sia INTE = 1). Il bit D0 viene controllato nel programma ed il dato è inserito dalla porta B, solamente quando D0 ($INTR_B$) va al valore logico 1.

Passo 3

Collegare il circuito mostrato nello schema (Fig. 6-21). Collegare il bit PC2 della porta C ($\overline{STB_B}$) all'uscita 1 del generatore d'impulsi.

Passo 4

Caricare il programma in memoria ed eseguirlo.

Passo 5

Impostare 01001001 sugli switch logici ed avviare e fermare il generatore d'impulsi. Che cosa si osserva sulla porta 0? Che cambiamento si vede sul contatore per dieci 7490?

Vediamo, come ci si aspettava, un'uscita 111 sul display a LED alla porta 0. Il contatore 7490, ha rilevato un solo conteggio. La lampada d'indicazione collegata a PC1 (IBF_B), è rimasta luminosa per tutto il tempo in cui il generatore di impulsi è rimasto in azione.

Programma (Fig. 6-22)

```
                      /
                      /
                      /PROGRAMMA D'INGRESSO PER IL PPI IN MODO 1
                      /
                      DB CNTRL 203
                      DB DATA3 202
                      DB DATA2 201
                      * 003 000
003 000 001                     LXIB     /CARICARE NELLA COPPIA DI REGISTRI D:
003 001 246                     246      /REGISTRO C: PAROLA DI CONTROLLO DEL
                                         /MODO
003 002 005                     005      /REGISTRO B: PAROLA DI CONTROLLO DI
                                         /SET BIT
003 003 171                     MOVAC    /CARICARE IN ACCUMULATORE LA
                                         /PAROLA DI CONTROLLO DEL MODO
003 004 323                     OUT      /INVIARLA AL
003 005 203                     CNTRL    /REGISTRO DI CONTROLLO
003 006 170                     MOVAB    /CARICARE IN ACCUMULATORE LA PAROLA
                                         /DI CONTROLLO SET/RESET BIT
```

```
003 007 323              OUT      /INVIARLA AL
003 010 203              CNTRL    /REGISTRO DI CONTROLLO
003 011 333      WAIT    IN       /INSERIRE IL BYTE DI STATO DELL'8255
                                  /DALLA
003 012 202              DATA3    /PORTA C
003 013 323              OUT      /INVIARE IN USCITA IL BYTE DI STATO
                                  /DELLA PORTA C PER VISUALIZZARLO
003 014 000              000
003 015 346              ANI      /MASCHERARE IL BIT D0: - INTRB
003 016 001              001
003 017 312              JZ       /INTRB = "0"?
003 020 011              WAIT     /SI, SALTARE A WAIT PER CARICARE I DATI
003 021 003              0        /NEL BUFFER D'INGRESSO DELLA PORTA B
                                  /DEL PPI
003 022 333              IN       /INTRB = "1", QUINDI INSERIRE I DATI ALLA
003 023 201              DATA2    /PORTA B
003 024 323              OUT      /INVIARE I DATI IN USCITA
003 025 000              000      /CODICE DISPOSITIVO DELLA PORTA 0
003 026 303              JMP      /SALTARE PER VEDERE SE CI SONO NUOVI
                                  /DATI
003 027 011              WAIT
003 030 003              0
```

Fig. 6-22. Programma per l'Esperimento 6-2.

Passo 6

Variare il valore impostato sugli switch logici ed avviare e fermare il generatore d'impulsi dopo ogni variazione. In ogni caso si dovrebbe rilevare che il dato, degli switch logici, compare alla porta 0.

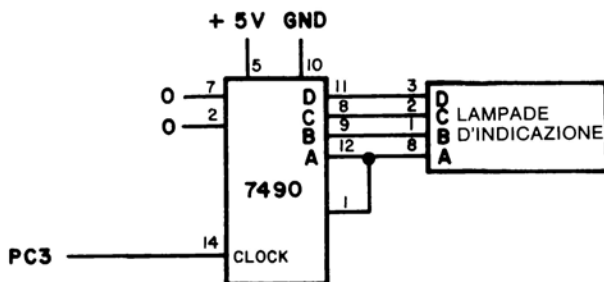



Fig. 6-23. Circuito contatore di una decade.

Passo 7

Fra quali passi di programma si ricicla, una volta che si è avviato il programma? Perchè?

Definizione del Segnale di Controllo d'Ingresso

$\overline{STB}$ (Ingresso Strobe)

Un valore "basso" su questo ingresso carica i dati nel latch d'fingresso.

IBF (Buffer d'Ingresso Pieno F/F)

Un valore "alto" su questa uscita indica che i dati sono stati caricati nel latch d'fingresso. In assenza, un riconoscimento IBF è posto a uno dal fronte di discesa dell'ingresso STB ed è azzerato dal fronte di salita dell'ingresso RD.

INTR (Richiesta Interrupt)

Un valore "alto" su questa uscita può essere usato per interrompere la CPU, quando un dispositivo d'ingresso sta richiedendo servizio.
INTR è posto a uno dal fronte di salita di STB se IBF e INTE sono "a uno". Esso viene azzerato dal fronte di discesa di $\overline{RD}$. Tale procedura consente ad un dispositivo d'ingresso, di richiedere servizio alla CPU segnalando semplicemente i propri dati alla porta.

INTE A

Controllato dal set/reset bti di PC4

INTE B

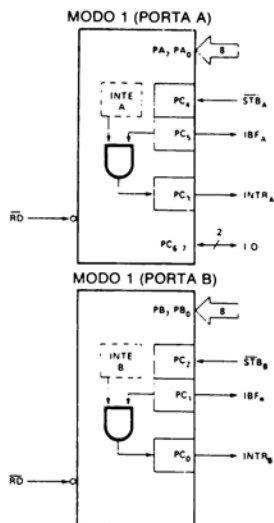
Controllato dal set/reset bid di PC2

Per gentile concessione della Interl Corp.

Fig. 6-24. Funzionamento delle porte A e B come porte d'ingresso in modo 1.

Il programma resta in loop fra gli indirizzi 003 011 e 003 017. Questo è il loop in cui si controlla il flag INTRB. Il programma uscirà da questo stato, quando il generatore d'impulsi sarà stato avviato e fermato e quando INTRB sarà stato posto al valore logico 1.

Variare gli switch impostando un nuovo valore ad 8 bit. Avviare il generatore d'impulsi e mantenerlo in azione. Succede niente quando esso viene avviato?

Fig. 6-25. Parola di stato della porta C per il funzionamento in modo 1 quando sia la porta A che la B sono porte d'ingresso.

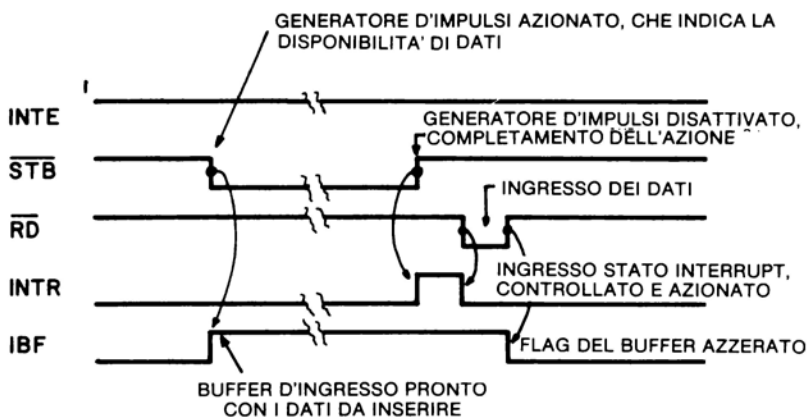



Fig. 6-26. Azione di avviamento ed arresto del generatore d'impulsi.

No, non succede niente fino a che il generatore d'impulsi non viene arrestato. Ciò è mostrato chiaramente in Fig. 6-26.

Passo 8

Che cosa succede quando si rilascia il generatore d'impulsi? Si abbandona il loop ovviamente. Che cosa succede ancora? Si sfrutti il diagramma di temporizzazione di Fig. 6-26 per aiutarsi nella spiegazione.

Si ha il trasferimento del nuovo schema ad 8 bit, sulle lampade d'indicazione sulla porta d'uscita.

Passo 9

Cambiare il byte al L0 indirizzo di memoria 002, con 004. Tale byte di controllo causa il reset al valore logico 0 dell'uscita del flip-flop interno, $INTE_B$. Eseguire nuovamente il programma. Azionare e fermare ripetutamente il generatore d'impulsi mentre si cambia l'impostazione degli switch logici. Si osserva qualche cambiamento alla porta 0 d'uscita?

Noi non ne vediamo.

Spiegare perchè la variazione di programma, agli indirizzi di memoria 003 002, ha un tale effetto. (*Suggerimento*: La documentazione Intel

specifica che "INTR è messo ad uno sul fronte di salita di $\overline{STB}$, se sia IBF che INTE sono al valore 1 logico").

Quando $INTE_B$ è uno 0 logico, il flag $INTR_B$ non è mai posto ad 1 con $\overline{STB_B}$, in tal modo non c'è alcuna possibilità, per il programma, di uscire dal loop WAIT.

Domande

1. Completare lo schema riportato in Fig. 6-27 per il funzionamento in modo 1 della porta A d'ingresso. Saranno necessari 8 switch logici

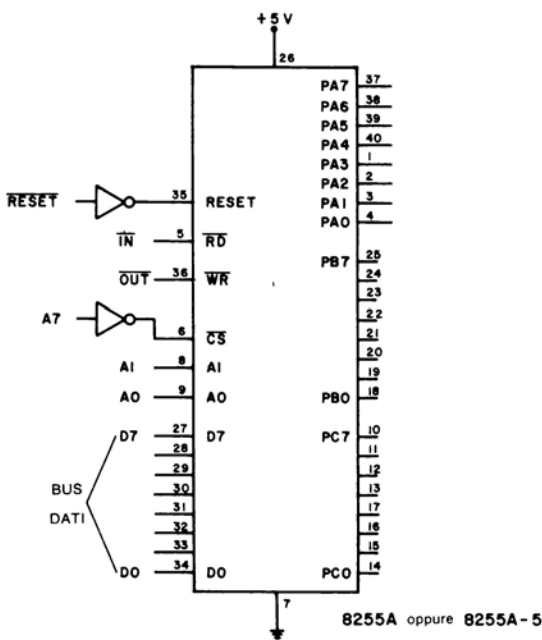



Fig. 6-27. Schema da completare.

sulla porta A, un generatore d'impulsi come strobe dell'ingresso $\overline{STB_A}$ e due lampade d'indicazione collegate ad IBF_A ed $INTR_A$.

2. Elencare, di seguito, le variazioni da apportare al programma dato in questo esperimento, per permettere di sfruttare la porta A nel funzionamento d'ingresso in modo 1.

L0 Indirizzo di Memoria	*Ingresso Porta A*	*Ingresso Porta B*
001		206
002		005
016		001
023		201

NOTA: Conservare il circuito impiegato in questo esperimento per il successivo.

ESPERIMENTO 6-3
FUNZIONAMENTO COMBINATO INGRESSO ED USCITA IN MODO 1 DEL PPI

Scopo

Lo scopo di questo esperimento è quello di mostrare il funzionamento in modo 1 dell'8255, per ingresso e uscita combinati, sfruttando la porta B come ingresso e la A come uscita. Si descrive anche il concetto di handshaking. Questo Esperimento è una sintesi dei procedimenti degli Esperimenti 6-1 e 6-2. Se si crede di aver fatto propri i procedimenti necessari per il funzionamento di I/O in modo 1 dell'8255, si può cercare di scrivere e verificare un programma che inserisca i dati dalla porta B (dagli switch logici) e li invii in uscita sulla porta A (alle lampade d'indicazione). Il listing del nostro programma è riportato in Fig. 6-28.

Passo 1

Collegare il circuito mostrato in Fig. 6-29. Si può anche impiegare un monitor del bus come mostrato in Fig. 6-30.

Passo 2

Caricare il programma in memoria.

Passo 3

Avviare il programma. Impostare 1111 1111 sugli switch logici. Il dato viene trasferito alle lampade di rivelazione? Perchè?

No, il dato non viene trasferito alle lampade di rivelazione della porta B, dal momento che il PPI non ha segnalato allo 8080 che il buffer d'ingresso, della sua porta B, è pieno.

Programma (Fig. 6-28)

```
                /
                /QUESTO  PROGRAMMA  INSERISCE  I  DATI  DALLA  PORTA  B
                /E LI INVIA IN USCITA ALLA PORTA A
                /
                DB PC2SET 005
                DB PORTA 200
                DB PORTB 201
                DB PORTC 202
                DB CNTRL 203
                *003 000
003 000 001            LXIB      /CARICARE NELLA COPPIA DI REGISTRI B:
003 001 246            246       /PAROLA DI CONTROLLO DEL MODO DEL
                                 /PPI
003 002 015            015       /PAROLA DI CONTROLLO SET BIT DEL PPI
                                 /(SET DI INTEA)
003 003 026            MVID      /CARICARE NELLA COPPIA DI REGISTRI D:
003 004 005            PC2SET    /PAROLA DI SET/RESET BIT DEL PPI
                                 /(SET DI INTEB)
003 005 171            MOVAC     /CARICARE IN A IL BYTE DI CONTROLLO
                                 /DEL MODO
003 006 323            OUT       /INVIARLO AL
003 007 203            CNTRL     /REGISTRO DI CONTROLLO DEL PPI
003 010 170            MOVAB     /CARICARE IN A IL SET DI PC6
003 011 323            OUT       /INVIARLO AL
003 012 203            CNTRL     /REGISTRO DI CONTROLLO DEL PPI
003 013 172            MOVAD     /CARICARE IN A IL SET DI PC2
003 014 323            OUT       /INVIARLO AL
003 015 203            CNTRL     /REGISTRO DI CONTROLLO DEL PPI
003 016 333     WAITB, IN        /INSERIRE LO STATO DEL PPI DALLA
                                 /PORTA C
003 017 202            PORTC
003 020 346            ANI       /MASCHERARE TUTTI I BIT
003 021 001            001       /ECCETTO IL BIT D0 (INTRB)
003 022 323            OUT       /INVIARE IL RISULTATO IN USCITA ALLA
                                 /PORTA 0
003 023 000            000
003 024 312            JZ        /INTRB = "0", QUINDI PROVARE DI NUOVO
003 025 016            WAITB
003 026 003            0
003 027 333            IN        /INTRB = "1", QUINDI INSERIRE I DATI
003 030 201            PORTB     /DALLA PORTA B
003 031 323            OUT       /INVIARE IL BYTE DATI IN USCITA ALLA:
003 032 200            PORTA     /PORTA A
003 033 333     WAITA, IN        /INSERIRE LA PAROLA DI STATO DEL PPI
                                 /DALLA
003 034 202            PORTC     /PORTA C
003 035 346            ANI
003 036 010            010       /MASCHERARE TUTTI I BIT ECCETTO
                                 /IL BIT D3 (INTRA)
003 037 312            JZ        /INTRA = "0", QUINDI PROVA DI NUOVO
003 040 033            WAITA
003 041 003            0
003 042 303            JMP       /INTRA = "1". DATI ACCETTATI ALLA
003 043 016            WAITB     /PORTA A QUINDI SALTA AD INSERIRE
003 044 003            0         /UN NUOVO BYTE DALLA PORTA B
```

Fig. 6-28. Programma per l'Esperimento 6-3.

Passo 4

Avviare e fermare il generatore d'impulsi # 1. Questo è il segnale di strobe dati, $\overline{STB_B}$, della porta B. Ciò causa il trasferimento dei dati? Perchè?

Sì. Tale azione fa da strobe dati per il buffer d'ingresso della porta B, causando la transizione al valore logico 1 del flag di buffer d'ingresso pieno della porta B. Il programma rileva ciò, inserisce i dati dalla porta B e li invia in uscita sulla porta A.

Schema del circuito (Fig. 6-29)

NOTA: Lo schema riporta i circuiti che si sono assemblati ed usati negli Esperimenti 6-1 e 6-2. Se tutti i collegamenti sono intatti, si può

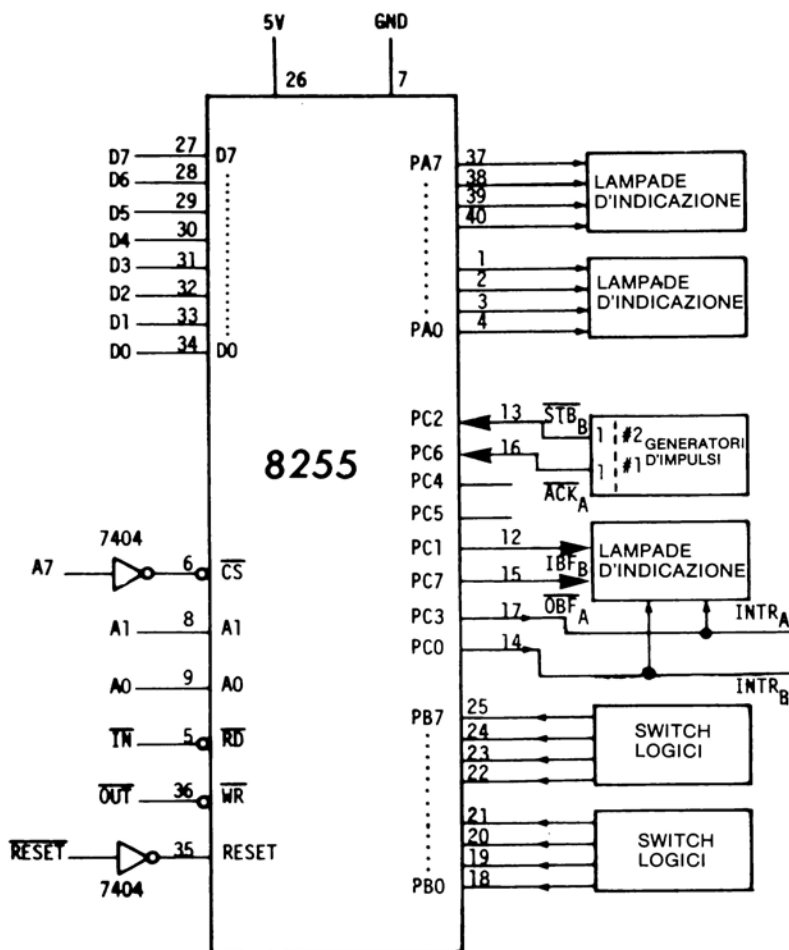



Fig. 6-29. Circuito per l'Esperimento 6-3.

passare ad esaminare il programma che viene fornito, oppure scrivere un proprio programma.

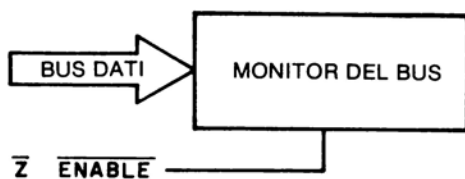



Fig. 6-30. Disposizione del monitor del bus.

Passo 5

Cambiare gli switch logici impostando 0000 0000. Avviare e fermare nuovamente il generatore d'impulsi # 1. Il dato viene trasferito? Perchè? Se non se ne è certi, si esamini il programma a WAITA.

No, il dato non viene trasferito, dal momento che il programma sta ora aspettando che il dispositivo d'uscita (lampade di indicazione) riconosca la ricezione del byte di dati.

Passo 6

Avviare e fermare il generatore d'impulsi # 2, cioè il segnale di controllo accettazione della porta A ($\overline{ACK_A}$). Che cosa si osserva ora alla porta A?

Si dovrebbe finalmente vedere che gli 0 sono stati trasferiti. Si ricordi, che si è appena riconosciuta la ricezione degli 1; quindi, avviare e fermare nuovamente il generatore d'impulsi # 2, per indicare che le lampade di rivelazione hanno ricevuto gli 0.

Passo 7

Cambiare ripetutamente l'impostazione degli switch e avviare e fermare i generatori d'impulsi # 2 ed # 1. Tali dati, vengono trasferiti ora?

Sì, dopo che si è completato un ciclo di attivazione dei generatori # 1 e # 2.

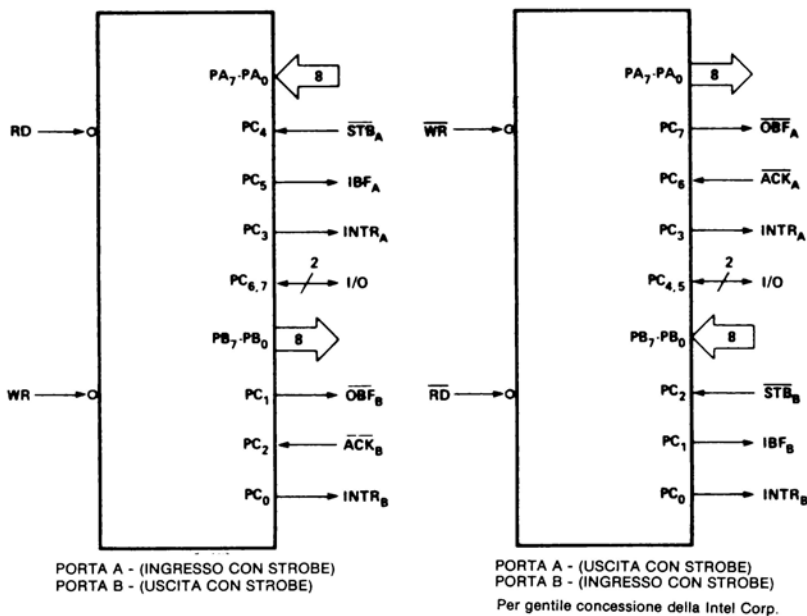



Fig. 6-31. Due combinazioni delle porte A e B in modo 1.

Passo 8

In questo passo, vengono mostrati gli schemi a blocchi (Fig. 6-31) per l'impiego di quattro delle possibili combinazioni del modo 1, per le porte A e B. Quali di questi schemi si riferisce a questo esperimento? Evidenziarlo con una risposta.

La situazione implementata in questo esperimento, è quella relativa allo schema sulla destra, che rappresenta una configurazione in modo 1, dal momento che la porta B è usata come ingresso (dagli switch logici) e la A come uscita (per le lampade di rivelazione).

Passo 9

Scrivere la forma della parola di stato in modo 1, per questo particolare esperimento. La parola di stato in modo 1 richiesta, può essere

costruita dalla Fig. 6-9, ricordando che il gruppo B è configurato come ingresso e quello A come uscita.

Il programma dato in questo esperimento, impiega due degli 8 bit per la parola di stato del modo 1. Quali bit sono usati? Perchè?

Nel loop WAITB del programma, si è visualizzato il bit D0 ($INTR_B$) della parola di stato in modo 1. Questo segnale va alto, quando i dati sono stati caricati nel buffer d'ingresso della porta B e sono pronti per l'ingresso nell'8080A. Nel loop WAITA, si è visualizzato il ($INTR_A$) della parola di stato in modo 1, dal momento che questo segnale va al valore logico 1, quando la periferica d'uscita alla porta A ha riconosciuto la ricezione del dato inviato alla porta stessa. Ciò viene eseguito in questo esperimento, avviando e fermando il generatore d'impulsi # 2 ($\overline{ACK_A}$).

Passo 10

In Fig. 6-32, è mostrata una serie di diagrammi di temporizzazione che descrive il funzionamento di questo programma.
Si esegua un reset del microcomputer e si avvii il programma in funzionamento single-step. $\overline{STB_B}$ è fornito dal generatore d'impulsi # 1 ed $\overline{ACK_A}$ dal generatore d'impulsi # 2. Si consideri il funzionamento single-step del programma, controllando IBF_B, $INTR_B$, $\overline{OBF_A}$ e $\overline{ACK_A}$. Attraverso quali passi il programma sta riciclando?

Il programma è nel loop WAITB, fra le locazioni di memoria 003 016 e 003 026, sta attendendo che $INTR_B$ vada alto. Tale situazione è descritta sulla estrema sinistra del diagramma di temporizzazione. Si passi attraverso il loop e ci si fermi al terzo ciclo di macchina, il ciclo d'ingresso, dell'istruzione d'ingresso. Ora si avvii e si fermi il generatore d'impulsi # 1, il segnale $\overline{STB_B}$ e si continui in single-step. Annotare di seguito ciò che si osserva.

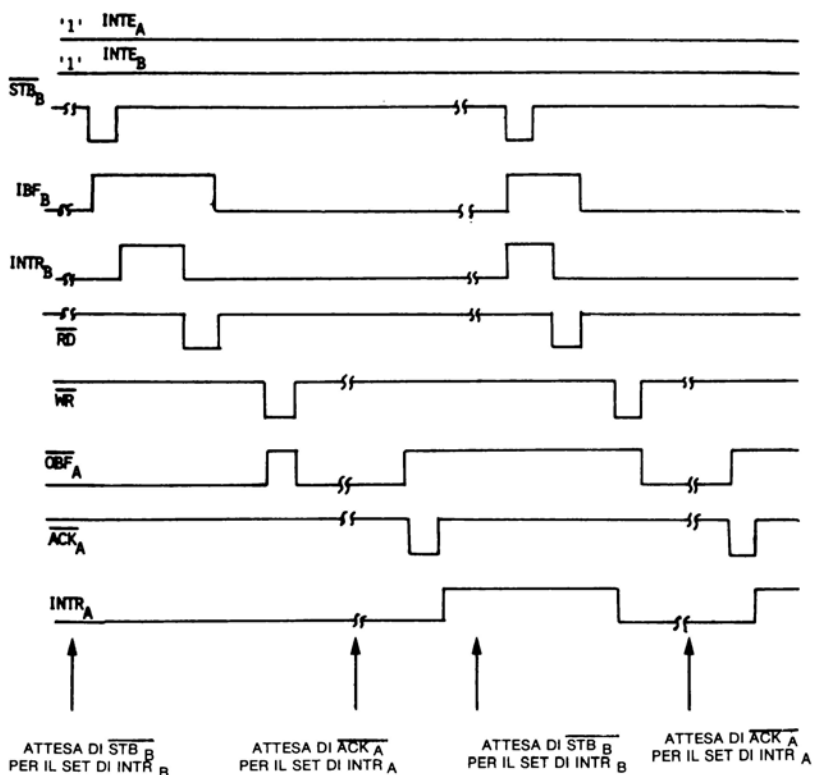



Fig. 6-32. Diagrammi di temporizzazione del programma.

Noi vediamo:

(a) IBF$_B$, è andato alto alla partenza del generatore di impulsi.
(b) INTR$_B$, è andato al valore di 1 logico, quando si è fermato il generatore d'impulsi. Entrambe queste azioni sono state rilevate nella parola di stato del modo 1 e sulle lampade di rivelazione.
(c) I dati, dagli interruttori logici, sono stati prelevati durante il terzo ciclo di macchina dell'istruzione IN alla locazione di memoria 003 030. INTR$_B$ viene qui azzerato.
(d) I flag IBF$_B$, vengono azzerati durante il *successivo* ciclo di macchina.

(e) Il programma, dopo aver inviato in uscita il byte sulla porta A, ricicla nel loop WAITA (locazioni di memoria da 003 033 fino a 003 041), aspettando che $INTR_A$ vada alto.

Ci si fermi ora al terzo ciclo di macchina dell'istruzione di ingresso alla locazione 003 034. Si vede la parola di stato del modo 1 sul monitor del bus. Avviare e fermare il generatore d'impulsi # 2. Quali cambiamenti si notano nella parola di stato del modo 1 e sulle lampade d'indicazione alla porta C?

Quando si è avviato il generatore d'impulsi, $\overline{OBF_A}$, è andato al valore logico 1. Quando si ferma, il generatore va al valore 1 $INTR_A$.

Si continui in single-step. Dove va il programma? Perchè?

Il programma entra nel loop WAITB, dopo essere uscito da WAITA, per inserire un byte dalla porta B.

Ripetere l'intero procedimento dello passo 10 e confermare il resto del diagramma di temporizzazione.

Passo 11

Si è qui usata, la procedura in single-step, per illustrare il funzionamento nel tempo del PPI per le operazioni di ingresso ed uscita in modo 1. In base al passo 10, si può suggerire un'importante applicazione della possibilità di single-step?

La possibilità di single-step, è molto utile nel debugging di un'interfaccia e del software di comando ad essa associato. Eseguendo in single-step il software e fermandosi al terzo ciclo di macchina delle istruzioni d'in-

gresso ed uscita, i byte che sono inseriti dall'interfaccia, quali lo stato ed i dati, possono essere controllati per ogni possibile errore. Allo stesso modo si possono controllare i byte di dati che sono inviati in uscita all'interfaccia.

NOTA: Conservare il circuito per l'esperimento successivo.

ESPERIMENTO 6-4
FUNZIONAMENTO DEL PPI IN MODO 1 CON INTERRUPT IN POLLING

Scopo

Lo scopo di questo esperimento, è quello di illustrare *il funzionamento in modo 1 con interrupt in polling* dell'8255.

Passo 1

Collegare i circuiti riportati in Fig. 6-35. Si noti che tutti i collegamenti del PPI sono uguali a quelli dell'Esperimento 6-3, ad eccezione del ricollegamento delle linee $INTR_A$ e $INTR_B$. Quindi, se si è eseguito quell'esperimento, si devono alterare solamente i collegamenti $INTR_A$ e $INTR_B$. Ci si assicuri che i pin PC2 ($\overline{STB_B}$) e PC6 ($\overline{ACK_A}$) del PPI, siano collegati alle uscite logiche 1 dei due generatori d'impulsi.

Passo 2

Caricare il programma nella memoria di lettura/scrittura del microcomputer.

Passo 3

Studiarsi il listing del programma. Si può vedere che ci sono quattro blocchi principali di istruzioni, di lunghezza variabile. Nello spazio di seguito elencare le locazioni di memoria di partenza, di ognuno di questi blocchi e descrivere brevemente le loro funzioni.

Programma (Fig. 6-33)

```
003 127 015        LOOP2, DCRC
003 130 302               JNZ
003 131 127               LOOP2
003 132 003               0
003 133 303               JMP
003 134 123               LOOP1
003 135 003               0
                   /      .
                   /      .
                   /      .
                   /      .
                   /      .
                   *003 150
003 150 365  SEARCH,      PUSHPSW  /INIZIO RICERCA DELLA SORGENTE
                                    /DI INTERRUPT
003 151 305               PUSHB
003 152 333               IN       /PAROLA DI STATO DELL'INGRESSO IN
                                   /MODO 1
003 153 202               PORTC
003 154 107               MOVBA    /SALVARLA
003 155 346               ANI      /MASCHERARE IL BIT D3 (INTRA)
003 156 010               010      /È LA PORTA A CHE STA INTERROMPENDO?
003 157 302               JNZ      /SI! - SALTARE ALLA ROUTINE DI
003 160 250               PASVCE   /SERVIZIO DELLA PORTA A
003 161 003               0
003 162 170               MOVAB    /NO, RIPRISTINARE LA PAROLA DI STATO
                                   /DEL MODO 1
003 163 346               ANI      /MASCHERARE IL BIT D0 = INTRB
003 164 001               001      /È LA PORTA B CHE STA INTERROMPENDO?
003 165 302               JNZ      /SI! - SALTARE ALLA ROUTINE DI SERVIZIO
003 166 210               PBSVCE   /DELLA PORTA B
003 167 003               0
003 170 301               POPB     /NO? - IGNORARE L'INTERRUPT
003 171 361               POPPSW
003 172 311               RET
                   /      .
                   /      .
                   /      .
                   /      .
                   /      .
                   *003 210
003 210 333        PBSVCE,IN       /INSERIRE I DATI DALLA PORTA B
003 211 201               PORTB
003 212 323               OUT      /INVIARLI IN USCITA ALLA PORTA 0
003 213 000               000
003 214 127               MOVDA
003 215 303               JMP      /SALTARE ALLA ROUTINE DI SERVIZIO
                                   /DELLA
003 216 250               PASVCE   /PORTA A
003 217 003               0
                   /      .
                   /      .
                   /
                   /      . .
                   /      .
                   *003 250
003 250 024        PASVCE,INRD     /INCREMENTARE D
003 251 172               MOVAD    /METTERE IL CONTENUTO DI D
                                   /NELL'ACCUMULATORE
003 252 323               OUT      /INVIARE IN USCITA IL CONTENUTO DI
                                   /D ALLA
003 253 200               PORTA    /PORTA A
003 254 301               POPB     /RIPRISTINARE LO STATO DEL
                                   /MICROCOMPUTER
```

Fig. 6-33. Programma per l'Esperimento 6-4 (segue).

```
003 255 361           POPPSW
003 256 373           EI          /ABILITAZIONE DEGLI INTERRUPT
003 257 311           RET
                DB MODE 246
                DB PORTA 200
                DB PORTB 201
                DB PORTC 202
                DB CNTRL 203
                DB PC6SET 015
                DB PC2SET 005
                *003 050
003 050 303           JMP         /IL PPI RICHIEDE SERVIZIO
003 051 150           SEARCH      /SALTARE ALLA RICERCA DELLA
003 052 003           0           /SORGENTE D'INTERRUPT
                /          .
                /
                /
                /          .
                /          .
                *003 100
003 100 061     START, LXISP      /INIZIARE L'INIZIALIZZAZIONE DEL PPI PER
003 101 377           377         /IL FUNZIONAMENTO IN MODO 1,CON
003 102 003           003         /INTERRUPT IN POLLING
003 103 076           MVIA        /CARICARE LA PAROLA DI CONTROLLO
                                  /DEL MODO 1
003 104 246           MODE        /PORTA A: USCITA; PORTA B: INGRESSO
003 105 323           OUT
003 106 203           CNTRL
003 107 076           MVIA        /SET DEL BIT D6 DEL PPI (INTEA)
003 110 015           PC6SET
003 111 323           OUT
003 112 203           CNTRL
003 113 076           MVIA        /SET DEL BIT D2 DEL PPI (INTEB)
003 114 005           PC2SET
003 115 323           OUT
003 116 203           CNTRL
003 117 373           EI          /ABILITARE L'8080A PER GLI INTERRUPT
003 120 001     WAIT,  LXIB       /INIZIARE L'ESECUZIONE DI UN
003 121 377           377         /LOOP D'ATTESA
003 122 377           377
003 123 005     LOOP1, DCRB
003 124 312           JZ
003 125 120           WAIT
003 126 003           0
```

Fig. 6-33. Programma per l'Esperimento 6-4.

Schema del circuito (Fig. 6-35)

I principali blocchi funzionali d'istruzioni, iniziano ai L0 indirizzi di memoria 100, 150, 210 e 250. Il primo blocco, è l'inizializzazione del PPI ed il loop d'attesa. Il secondo, è relativo al polling del PPI per determinare la sorgente di interrupt. I blocchi tre e quattro, sono rispettivamente le routine di servizio delle porte B e A. La subroutine di servizio della porta B, inserisce i dati degli switch-logici dalla porta d'uscita B alla porta 0, incrementa il byte di dati nel registro D ed invia il risultato in

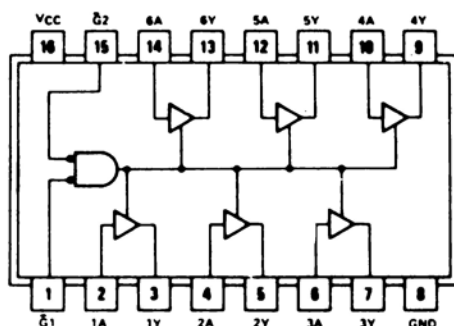
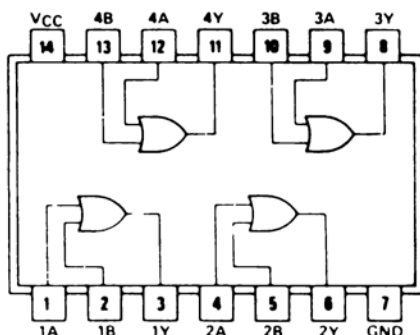
Configurazione del pin del chip circuito integrato (Fig. 6-34)

(A) 8095.

(B) 7432.

Fig. 6-34. Configurazione dei pin degli IC.

uscita sulla porta A. La subroutine di servizio di questa porta (PASVCE), incrementa il contenuto del registro D ed invia il risultato in uscita sulla porta A.

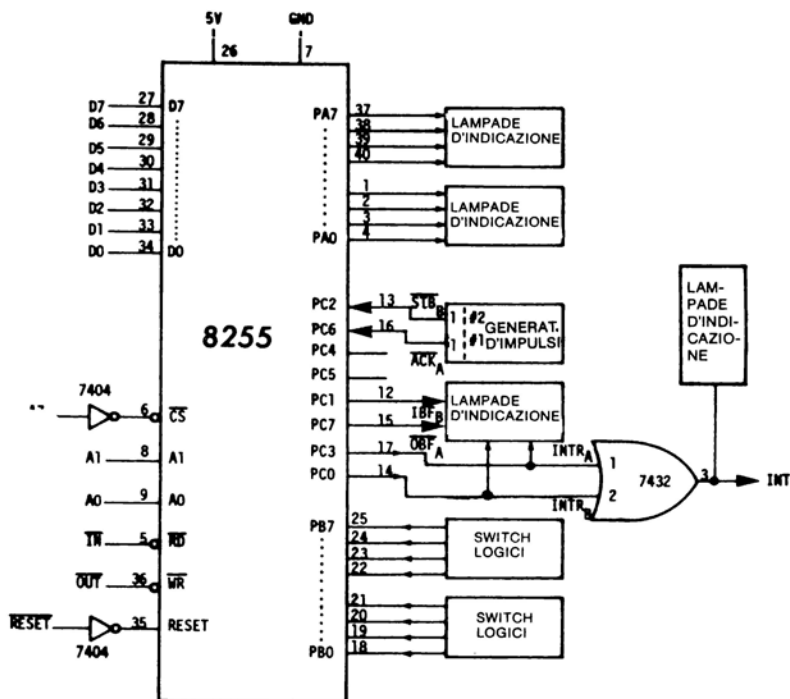
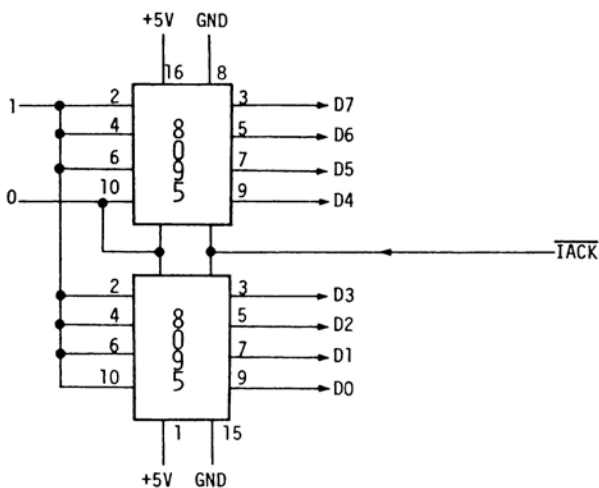
Durante lo studio del programma si possono notare due caratteristiche. La prima è che i codici usati per indirizzare le porte A, B, C e il registro di controllo del PPI, sono gli stessi usati negli Esperimenti 6-1 e 6-3, dal momento che i collegamenti delle linee di controllo $\overline{CS}$, A1 e A0, dell'8255, in questo esperimento sono gli stessi usati nei precedenti. La seconda è che, poichè questo è un esperimento di interrupt, il flag di abilitazione interrupt dell'8080A deve essere posizionato prima che il microcomputer possa essere interrotto. Il flag viene messo a 1, mediante l'istruzione EI, *dopo* che il PPI è stato inizializzato, per far sì che $INTR_A$ e $INTR_B$ siano entrambi al valore logico 0.

(A) Connessioni del PPI

(B) Circuiti buffer 3-state

Fig. 6-35. Circuiti per l'Esperimento 6-4.

Passo 4

Eseguire il programma iniziando dall'indirizzo 003 100. La linea INT, è al valore logico 1? Se no, perchè?

La linea d'interrupt, INT, dovrebbe essere al valore logico 0. Un 1 logico indica che $INTR_A$ o $INTR_B$, sono al valore logico 1, quindi c'è richiesta di servizio per le porte A o B. Ciò può avvenire solamente se si sono avviati i generatori d'impulsi $\overline{STB_B}$ o $\overline{ACK_A}$.

Passo 5

Impostare gli switch logici a 0. Ora avviare e fermare il generatore d'impulsi # 2 ($\overline{STB_B}$). Descrivere ciò che si vede nello spazio sotto e metterlo in relazione con ciò che ci si aspetta dal programma.

Avviare $\overline{STB_B}$, significa caricare la porta B con il byte di dati degli switch logici. Quando si arresta il generatore di impulsi $INTR_B$ e quindi INT, vengono messi ad uno. Ciò causa l'interruzione del microcomputer. Uno 0 logico su IACK, accettando l'interrupt, fa sì che il byte 357 o RST5, siano "spinti" nel registro d'istruzione. Il controllo del programma è vettorizzato per mezzo di salti a 000 050 e 003 050 alla locazione 003 150 dove, attraverso polling del PPI, si stabilisce che ad interrompere è la porta B. La routine di servizio di questa porta, invia in uscita il contenuto della porta stessa prima alla porta 0 e poi, dopo un incremento, alla A. I byte di dati 000 e 001, dovrebbero quindi essere osservati rispettivamente alla porta 0 e alla A.

Passo 6

Avviare e bloccare il generatore d'impulsi $\overline{ACK_A}$. Descrivere, nello spazio sotto, cosa si osserva ogni volta che il generatore d'impulsi viene avviato e bloccato.

Abbiamo visto che il contenuto della porta A, viene incrementato ogni volta che il generatore d'impulsi è azzerato.

Passo 7

Collegare il circuito di monitor del bus, al bus dei dati con il suo ingresso, abilitazione latch, connesso al valore logico 0, in modo che il monitor sia continuamente abilitato. Eseguire il programma in single-step. Avviare il generatore d'impulsi $\overline{STB}_B$ ed osservare cosa succede a IBF_B. Fermare $\overline{STB}_B$ e vedere cosa avviene a INT. Iniziare ora il single-step attraverso il programma. Dopo che il microcomputer avrà completato l'istruzione corrente, si vedrà spingere nello stack l'indirizzo dell'istruzione, che sarà eseguita al ritorno dall'interrupt, l'istruzione RST5 sul bus dati, e quindi l'esecuzione del programma come descritto nel passo 5. Annotarsi i byte che compaiono sul bus dati, confrontarli con il listing del programma. Notare le variazioni di $INTR_A$. Il diagramma di temporizzazione è uguale a quello riportato in Fig. 6-32. Ripetere l'esercizio quando ACK_A è avviato. Prendere nota delle variazioni delle linee di stato e di controllo del PPI. Per la comprensione del funzionamento del circuito, si troveranno utili le Fig. 6-4 e 6-6.

NOTA: Conservare il programma ed il circuito per il prossimo esperimento.

ESPERIMENTO 6-5
FUNZIONAMENTO DEL PPI IN MODO 1 CON INTERRUPT VETTORIZZATO

Scopo

Lo scopo di questo esperimento è quello di descrivere il *funzionamento in modo 1 con interrupt vettorizzato* del PPI.

Schema del circuito (Fig. 6-36)

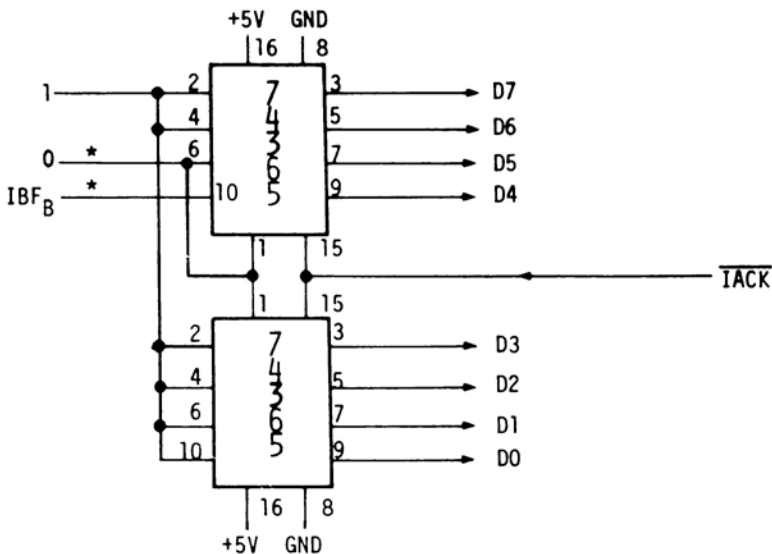



Fig. 6-36. Circuito per l'Esperimento 6-5.

Programma (Fig. 6-37)

```
                DW PASVCE 003 250
                DW PBSVCE 003 210
                *003 010
003 010 365             PUSHPSW   /MEMORIZZARE LO STATO DEL
                                  /PROGRAMMA
003 011 305             PUSHB
003 012 303             JMP       /SALTARE ALLA SUBROUTINE DI SERVIZIO
003 013 250             PASVCE    /DELLA PORTA A
003 014 003             0
                /
                /
                /
                /
                /
                *003 030
003 030 365             PUSHPSW   /MEMORIZZARE LO STATO DEL
                                  /PROGRAMMA
003 031 305             PUSHB
003 032 303             JMP       /SALTARE ALLA SUBROUTINE DI SERVIZIO
                                  /DELLA
003 003 210             PBSVCE    /PORTA B
003 034 003             0
```

Fig. 6-37. Programma per l'Esperimento 6-5.

Passo 1

Il circuito di questo esperimento, è uguale a quello usato nell'esperimento precedente, ad eccezione di due variazioni che sono state fatte sugli ingressi dei buffer three-state 8095 (SN74365) e che sono stati indicati con un asterisco. Eseguire le variazioni riportate nello schema di Fig. 6-36.

Passo 2

Le routine di servizio delle porte A e B e i codici d'inizializzazione del PPI, che si caricavano nell'esperimento 6-4, verranno usate anche in questo. Aggiungere in memoria le istruzioni del programma il cui listing è in Fig. 6-37. Si noti che le istruzioni nelle locazioni da 003 050 a 003 052 e da 003 150 a 003 172, non verranno usate in questo programma.

Passo 3

Il circuito buffer three-state 8095, mostrato nello schema, è impiegato per generare le due istruzioni di restart di vettore che sono spinti nel registro d'istruzione, durante un interrupt della porta A o B. Per generare due istruzioni di restart di vettore, la linea di stato IBF_B, della porta B, è connessa al buffer istruzione di interrupt. Con riferimento alle Fig. 6-4 e 6-6, ricavare gli stati logici di IBF_A *durante un interrupt*. Completare la tabella 6-1, quindi ricavare le istruzioni di restart che vengono spinte nel registro d'istruzione, durante l'ingresso di dati dalla porta B e l'uscita dati dalla A.

Tabella 6-1. Dati istruzione restart

	IBF_A	Istruzioni RST	
		Binario	Ottale
Porta A ingresso			
Porta B uscita			

Durante l'ingresso alla porta B, IBF_B è al valore logico 1 e si genera un'istruzione RST3 (337). Durante l'uscita alla porta A, IBF_B, è al valore logico 0 e si genera un'istruzione RST1 (317).

Passo 4

Eseguire il programma iniziando dalla locazione 003 100.

Passo 5

Impostare gli switch logici a 377. Avviare e bloccare il generatore d'impulsi $\overline{STB_B}$. Descrivere e spiegare che cosa si osserva.

Abbiamo visto che il byte di dati 377, viene inviato in uscita alla porta 0 MMD - 1 ed il byte di dati 000 viene inviato in uscita alle lampade di indicazione alla porta A. Ciò accade perchè il funzionamento del microcomputer è vettorizzato direttamente alla routine di servizio della porta B, le cui operazioni si sono descritte nel passo 5 dell'Esperimento 6-4.

Passo 6

Ora avviare e fermare il generatore d'impulsi $\overline{ACK_A}$ parecchie volte. Cosa si osserva?

Abbiamo visto che il byte di dati sulle lampade d'indicazione della porta A, viene incrementato di uno ogni volta che il generatore d'impulsi viene avviato e bloccato.

Passo 7

Commutare il microcomputer in funzionamento single-step. Avviare e fermare il generatore d'impulsi $\overline{ACK_A}$. Passare ora in single-step attraverso l'esecuzione del microcomputer dell'interrupt della porta A. Osservare l'istruzione RST 1 (317) sul bus dati, impiegando un monitor del bus che sia permanentemente abilitato. Annotarsi gli stati logici di $\overline{OBF_A}$ e INT passando attraverso il programma e confrontare quei valori con quelli di Fig. 6-6. Per conservare la linea di esecuzione del programma, tornerà utile scrivere, nello spazio qui sotto, i byte che si osservano sul bus dati e confrontare questi con il programma.

I primi pochi byte che abbiamo visto noi, sono i seguenti:

```
***  Completamento dell'istruzione corrente
317  Istruzione RST 1
003  HI byte d'indirizzo del contatore di programma
127  L0 byte d'indirizzo del contatore di programma
303  Istruzione di salto in esadecimale a 000 010
010
003
365
 .
 .
 .
ecc.
 .
 .
 .
```

CAPITOLO 7

FUNZIONAMENTO IN MODO 2: I/O BIDIREZIONALE

7-1. INTRODUZIONE

Con l'introduzione dei microcomputer, l'implementazione di sistemi di controllo di processo digitali, impieganti un certo numero di computer, è divenuto più comune. In questo tipo di sistema di controllo a microcomputer *distribuito*, diversi piccoli computer - sempre in maggior numero - sono dedicati al controllo di piccole parti dell'intero processo. In un processo chimico, ad esempio, per controllare una parte di esso, si può usare un microcomputer, come ad esempio la pressione o la temperatura di una colonna di distillazione. Con microcomputer usati in questo modo, viene poi impiegato un computer più grande, con velocità, memoria e memoria a disco maggiori (un minicomputer o un più grande e più potente microcomputer), come *supervisore* al funzionamento dell'intero processo, per controllare ed alterare con opportuni comandi, le operazioni dei controllori a microcomputer dedicati. Ciò porta, naturalmente, al concetto di gerarchia dei computer di controllo. Nell'esempio citato il computer supervisore era ad un livello più alto dei microcomputer dedicati. In conseguenza di quanto detto, il supervisore di piccoli computer o microcomputer, è spesso indicato quale *master* (padrone), mentre i microcomputer dedicati, a livello inferiore, vengono chiamati *slave* (schiavi).

Il concetto master-slave, è illustrato in Fig. 7-1, la quale indica anche il ruolo che può giocare il PPI in questa situazione. Il requisito essenziale di un sistema di controllo a microcomputer distribuito, è il trasferimento dei dati nei due sensi fra il microprocessore master e le unità slave (MPU). Tali dati possono essere sotto forma di *blocchi di dati*, che vengono raccolti dallo slave e trasferiti al master per l'elaborazione; oppure possono essere *dati di un punto prestabilito*, che vengono passati dal master allo slave in modo che esso possa mantenere un parametro del

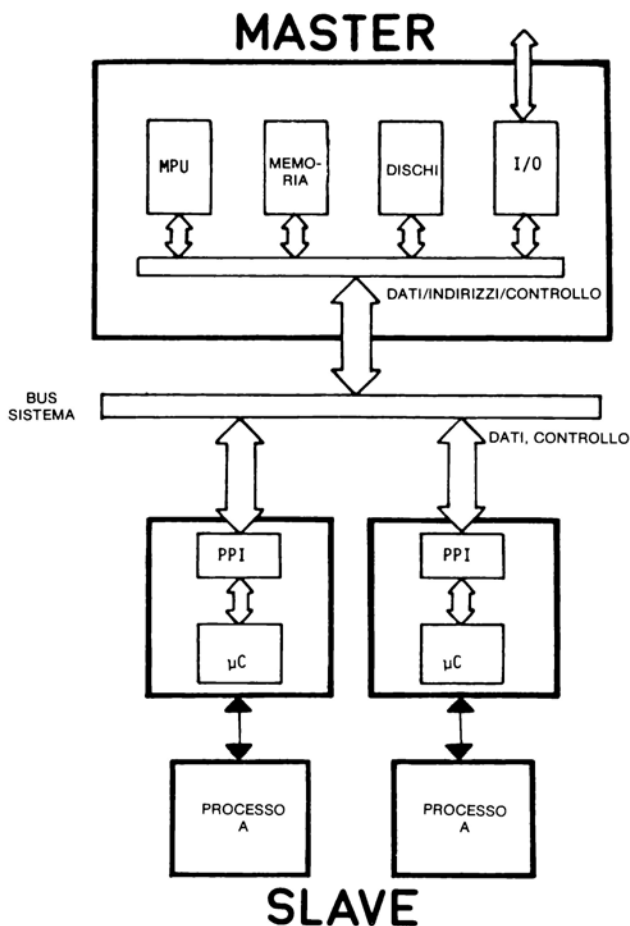



Fig. 7-1. Computer distribuito per il controllo di processo che illustra il modo in cui il PPI viene impiegato come elemento d'interfaccia fra i bus bidirezionali delle MPU master e slave.

processo (temperatura, pressione, spessore, etc.) ad un determinato valore. Quindi la MPU slave, deve essere collegata ai bus di controllo e dei dati della MPU master.

Il collegamento di una MPU master con diverse MPU slave, deve essere fatto con cura per assicurare un trasferimento dati valido e per evitare una situazione in cui una MPU slave sovraccarichi il bus dati della MPU master. Per tale motivo è necessario un circuito d'interfaccia fra il master e *ogni* MPU slave. Il PPI, nel suo funzionamento in modo 2,

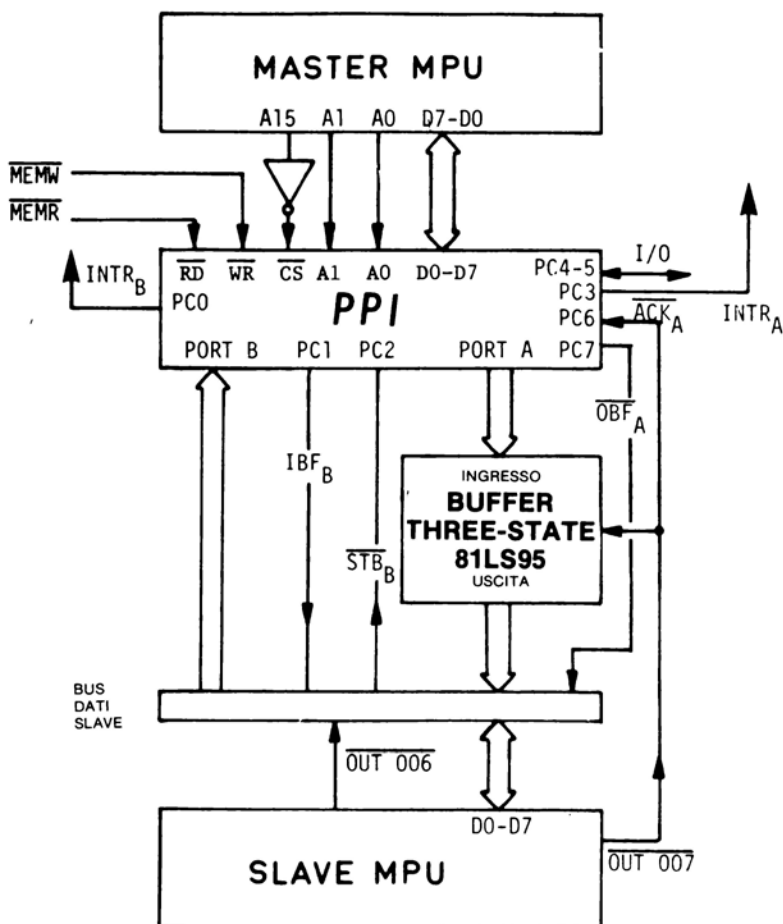
fornisce l'interfaccia richiesta. I requisiti che un tale circuito d'interfaccia deve avere, possono essere riassunti come segue:

- Deve permettere un *flusso bidirezionale di dati.*
- Poichè il trasferimento dati al e dal master può avvenire in ogni istante, sono necessari *segnali di handshaking* per assicurare un ordinato flusso di dati fra il master e lo slave.
- Dal momento che i bus dati della MPU master e degli slave devono essere interfacciati, sono necessari buffer three-state fra l'MPU master e il PPI (per far sì che l'MPU slave non sovraccarichi il bus dati dell'MPU master) e fra il PPI e l'MPU slave (per assicurare che le linee dati del master non sovraccarichino il bus dati dell'MPU slave). Il primo buffer three-state, sul lato del PPI verso l'MPU master, è abilitato dalla linea di selezione chip del PPI. Il secondo buffer three-state, sul lato del PPI verso lo slave, è abilitato da un segnale di handshaking dall'MPU slave quando è pronto ad accettare dati dal master.
- I dati che devono essere trasferiti dal master all'MPU slave, devono subire *latch* da parte del circuito di interfaccia e devono essere trattenuti fino a che l'MPU slave non sia pronta ad accettarli.

7-2. FUNZIONAMENTO DEL PPI IN MODO 1 PER UN FLUSSO BIDIREZIONALE DI DATI

La Fig. 7-2, mostra un possibile schema di impiego del PPI nella configurazione 1, per implementare l'interfaccia necessaria fra MPU master e slave per I/O bidirezionale. Occorre sottolineare il fatto che il *PPI non è normalmente configurato in questo modo* per fornire I/O bidirezionale fra due MPU. La ragione per cui si è riportato lo schema, è quella di chiarire i requisiti dell'interfaccia ed introdurre le caratteristiche del funzionamento in modo 2.

Con riferimento poi alla Fig. 7-2, il PPI è stato configurato con la porta A usata per l'uscita dati (e quindi per trasmettere dati allo slave) e la B è usata come ingresso (e quindi per ricevere dati dall'MPU slave). Consideriamo prima il *trasferimento di dati dallo slave all'MPU master.* Quando lo slave è pronto a trasmettere dati al master, esso deve scriverli nella porta B del PPI, che, per lo slave interessato, diventa un'unica porta *d'uscita* in handshaking. Mediante un impulso, attivo basso, di selezione dispositivo, sulla linea di strobe $\overline{STB_B}$, lo slave fa sì che i propri dati subiscano latch nella porta B del PPI. Quest'ultimo risponde nel solito modo, alzando la propria linea IBF_B per accettare la ricezione dati dallo slave e per segnalare al master che un byte di dati è disponibile per

Fig. 7-2. Illustrazione di come si potrebbe programmare il PPI in modo 1 per fornire I/O bidirezionale fra MPU master e slave.

l'ingresso. Una volta che il master ha letto il byte di dati dell'MPU slave dalla porta B, il PPI abbassa la propria linea IBF_B, per segnalare allo slave che il primo byte di dati è stato trasferito e che la porta B può essere caricata con un nuovo byte di dati. Il trasferimento dati appena descritto è semplicemente un'operazione d'ingresso alla MPU master, in handshaking in modo 1, impiegando il PPI per la sincronizzazione dei dati. L'unica differenza fra questa descrizione e quella del Paragrafo 6-2, dove si tratta l'ingresso in modo 1, è che la sorgente dei dati in questo caso, è un secondo microcomputer.

Consideriamo ora, il *trasferimento dei dati dal master allo slave*. Si usa un'operazione d'uscita del PPI attraverso la porta A, Fig. 7-2. Ad ogni modo, le linee dati della porta A, non possono essere collegate direttamente al bus dati dell'MPU slave, dal momento che la porta A del PPI appare come *ingresso* periferico all'MPU slave e sovraccaricherebbe il suo bus dati. La soluzione, come in tutte le operazioni d'ingresso del microcomputer, è quella d'impiegare un buffer three-state ad 8 bit fra la porta A e il bus dati dell'MPU slave. Si usano poi le linee di handshaking del modo 1, per sincronizzare il trasferimento dati dal master allo slave come segue. Innanzi tutto il master invia un byte di dati alla porta A del PPI nel solito modo. Questa operazione, fa sì che si abbassi il flag $\overline{OBF_A}$, per indicare che i dati hanno subito latch alla porta A. Quando l'MPU slave rileva l'$\overline{OBF_A}$ basso, inserisce i dati presenti sulla porta A abilitando il buffer three-state, che in Fig. 7-2 si trova fra il PPI e l'MPU slave, con un impulso di selezione dispositivo. Si noti che quest'ultimo impulso di selezione dispositivo. Si noti che quest'ultimo impulso è anche usato come strobe $\overline{ACK_A}$, che quindi avvisa il PPI che i dati alla porta A sono stati ricevuti dallo slave. Con $\overline{OBF_A}$, ora alto in seguito all'impulso $\overline{ACK_A}$, il master è libero di impegnare la porta A con un altro byte da trasferire all'MPU slave.

Quindi le caratteristiche *hardware* principali di questa interfaccia master/slave sono:

- La necessità di un buffer three-state fra la porta e il bus dati dell'MPU slave, dal momento che, non appena lo slave è interessato, il trasferimento dati sopra descritto, è una semplice operazione d'ingresso in handshaking.
- L'impiego dei segnali di handshaking della porta A per sincronizzare ancora il flusso di dati.
- La necessità di collegare il PPI al microcomputer master come un dispositivo I/O. Nell'illustrazione in Fig. 7-2, il PPI era collegato per I/O in mappa di memoria.

Come generalizzazione finale di questo discorso, si deve notare che per il buon esito dei trasferimenti dati master/slave, sia il master che lo slave devono controllare i flag IBF e $\overline{OBF}$. I dettagli di quanto detto sono trattati nel Paragrafo 7-4. Guardiamo prima come la configurazione del PPI in modo 2, della porta A, permetta gli stessi trasferimenti dati, master/slave, descritti sopra.

7-3. CARATTERISTICHE DEL MODO 2 DEL PPI

In Fig. 7-3 è indicata l'assegnazione delle linee delle porte da A a C, quando la porta A è configurata per il funzionamento in modo 2. La A è ora una porta I/O bidirezionale in coppia con cinque linee della porta C (da PC3 a PC7) come linee di controllo handshaking. La porta B può essere configurata come semplice I/O in modo 0 (nel qual caso le linee della porta C PC0-PC2 sono disponibili come ingressi/uscite), oppure può essere programmata come ingresso o uscita con handshaking in modo 1 (nel qual caso le linee della porta C PC0-PC2, hanno la loro solita assegnazione come linee di controllo handshaking). Guardiamo ora più da vicino la porta A. La Fig. 7-4 mostra la disposizione funzionale della porta A quando il PPI è programmato per il funzionamento in modo 2. La A è ora una vera porta bidirezionale ed è progettata per fornire un'interfaccia fra due microcomputer o fra un microcomputer ed una periferica che trasmette e riceve dati. Una unità floppy-disc, è un esempio di questo tipo di periferica, sebbene essa sia solitamente interfacciata ad un microcomputer per mezzo di un circuito integrato dedicato per controllo di floppy-disc, quale ad esempio il Western Digital DM 1771. Si confrontino ora le linee del PPI di Fig. 7-2 con quelle del PPI in modo 2 che sono illustrate in Fig. 7-4 e che vengono usate per implementare una struttura I/O bidirezionale equivalente. In Fig. 7-4 si supponga

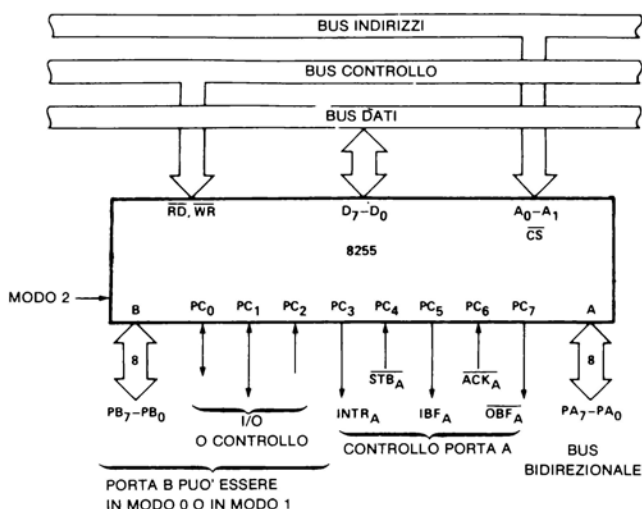



Fig. 7-3. Assegnazione delle linee d'interfaccia del PPI quando la porta A è programmata per il funzionamento in modo 2.

(A) Schema a blocchi della porta A

(B) Parola di controllo del modo

Fig. 7-4. Schema a blocchi funzionale per la porta A quando il PPI è programmato con la parola di controllo di modo per il funzionamento in modo 2.

che, come in Fig. 7-2, il PPI sia collegato all'MPU master e che questo sia quindi sulla sinistra del PPI. Nell'esempio di implementazione del modo 1 di Fig. 7-2, erano necessarie due porte del PPI, A e B, per effettuare un trasferimento di dati nelle due direzioni fra l'MPU master e l'MPU slave. Nel funzionamento del PPI in modo 2 (Fig. 7-4), la porta A è configurata come una vera porta bidirezionale per il trasferimento dei dati fra il master e lo slave, la porta B resta così libera per eventuali compiti di

ingresso/uscita, in modo 0 o in modo 1 associati all'MPU master. Si può inoltre vedere che, in entrambe le figure, per sincronizzare i trasferimenti di dati, s'impiegano gli *stessi* set di segnali handshaking. La configurazione del PPI in modo 2, quindi, è semplicemente un concentrato del modo 1, implementato con l'interfaccia I/O bidirezionale mostrata in Fig. 7-2, in una sola porta (A) per il trasferimento dei dati e cinque linee della porta C (PC7 - PC3) per l'handshaking.

Vediamo ora la funzione di ognuno dei segnali di controllo handshaking del modo 2. Per *l'ingresso dei dati al PPI in modo 2* (ciò è equivalente al trasferimento dati dall'MPU slave al master di Fig. 7-2):

- $\overline{STB_A}$, linea ingresso *strobe*, carica i dati dall'MPU slave nel latch d'ingresso della porta A, quando è a un valore logico basso.
- IBF_A, flag di *buffer d'ingresso pieno*, è un segnale di accettazione attivo alto, il quale indica che i dati sono stati caricati nel latch d'ingresso del PPI.

L'ingresso dei dati al PPI in modo 2, attraverso la porta A, è identico a quello in modo 1. I segnali di handshaking del modo 2, $\overline{STB_A}$ e IBF_A, sono identici in modo e funzione ai segnali di handshaking dell'ingresso in modo 1, trattati nel Capitolo 6 ed impiegati in Fig. 7-2. La differenza principale fra le operazioni d'ingresso in modo 2 ed in modo 1, sta nel tipo di dispositivo che dovrà essere collegato alle linee della porta A del PPI. Nel modo 2 alla porta A, è collegato un dispositivo capace sia di fornire che di ricevere dati. Nel modo 1 alla porta A, dovrà essere collegata una periferica che fornisca solamente dei dati, come ad esempio un convertitore A/D.

Per *l'uscita dati dal PPI in modo 2* (ciò è equivalente al trasferimento dati dal master allo slave di Fig. 7-2):

- $\overline{OBF_A}$, flag di *buffer d'uscita pieno*, va al valore logico 0 quando l'MPU master ha scritto i dati nel latch d'uscita della porta A.
- $\overline{ACK_A}$, l'ingresso di *accettazione*, è portato ad un valore logico basso dall'MPU slave per segnalare al PPI che si stanno leggendo i dati nel latch alla porta A. Ciò abilita il buffer three-state d'uscita della porta A. In condizioni normali $\overline{ACK_A}$ è, di solito, ad un valore logico alto ed il buffer d'uscita della porta A è tenuto nel suo stato di alta impedenza.

La differenza importante fra le configurazioni d'uscita del modo 2 e del modo 1 della porta A, è il buffer d'uscita three-state incorporato, il quale nel funzionamento in modo 2, è abilitato da un valore logico basso sulla linea $\overline{ACK_A}$. Questo è necessario poichè il dato mantenuto nel latch

d'uscita del PPI è un *ingresso* dell'MPU slave e deve essere con buffer per evitare un sovraccarico del bus dati dell'MPU slave. Quindi la conseguenza dell'impiego del funzionamento in modo 2 del PPI, quale interfaccia fra un MPU master e uno slave, è di liberare la porta B e le linee di controllo da PC0 a PC2 per altri compiti e di eliminare la necessità di un buffer three-state esterno. La porta A, nel funzionamento in modo 2, funziona come un buffer di dati fra i microcomputer master e slave.

Le rimanenti funzioni di Fig. 7-4, indicate con INTE 1, INTE 2, $INTR_A$, sono relative al modo in cui il PPI avvisa l'MPU master che si è completata una trasmissione master-slave, o che si sono ricevuti dei dati dallo slave. Nel primo caso, $\overline{OBF_A}$ va alto per indicare che si è completata una trasmissione dati da master a slave. Nel secondo, va alto IBF_A, per indicare che il PPI ha ricevuto dei dati dallo slave. In entrambi i casi $INTR_A$, flag di *richiesta interrupt* (PC3 in Fig. 7-4), va al valore logico 1, purchè siano alti anche INTE 1 (nel caso di $\overline{OBF_A}$) o INTE 2 (nel caso di IBF_A). INTE 1 ed INTE 2, flag di *abilitazione interrupt*, sono flip-flop interni, la cui funzione di controllo della generazione di interrupt da parte rispettivamente di $\overline{OBF_A}$ e IBF_A, è simile a quella dei flag di abilitazione interrupt del modo 1, $INTE_A$ e $INTE_B$. I flag INTE 1 e INTE 2,

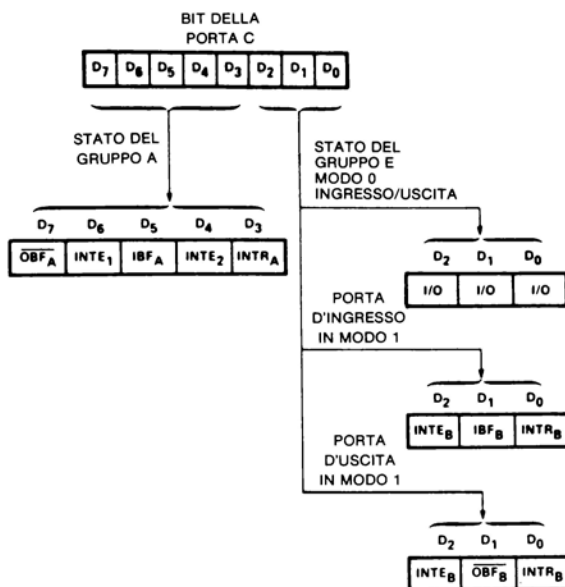



Fig. 7-5. Parola di stato del modo 2.

2, sono controllati dall'MPU master tramite le operazioni di set/reset bit, rispettivamente sui bit PC6 e PC4 del PPI. Quindi la linea $INTR_A$, può essere usata per interrompere l'MPU master sia per il controllo di operazioni d'ingresso che per l'uscita e, come detto nel Capitolo 6, può essere impiegata sia per interrupt vettorizzato che in polling. Con interrupt vettorizzato, $\overline{OBF}_A$ e IBF_A, possono essere collegati ad un registro istruzione interrupt, per generare vettori unici di istruzioni di restart al master, per operazioni d'ingresso ed uscita. Quando si usa interrupt in polling, l'MPU master deve determinare se deve essere servita un'operazione d'ingresso o di uscita, controllando rispettivamente, lo stato del flag IBF_A ed $\overline{OBF}_A$. Ciò viene fatto leggendo la porta C, la quale fornisce la parola di stato del modo 2, indicata in Fig. 7-5. I bit D7-D3, forniscono lo stato della porta A bidirezionale. Si noti che l'assegnazione di D6 e D4 ad INTE 1 ed INTE 2, è coerente con le operazioni di set/reset bit su PC6 e PC4, le quali sono richieste per posizionare rispettivamente, i flag interni INTE 1 o INTE 2.

7-4. FUNZIONAMENTO E CONDIZIONI DEL MODO 2 DEL PPI

(A) Requisiti hardware

La Fig. 7-6, mostra un'interconnessione tipica di due microcomputer, che impiega il PPI come elemento d'interfaccia. Il PPI, è collegato

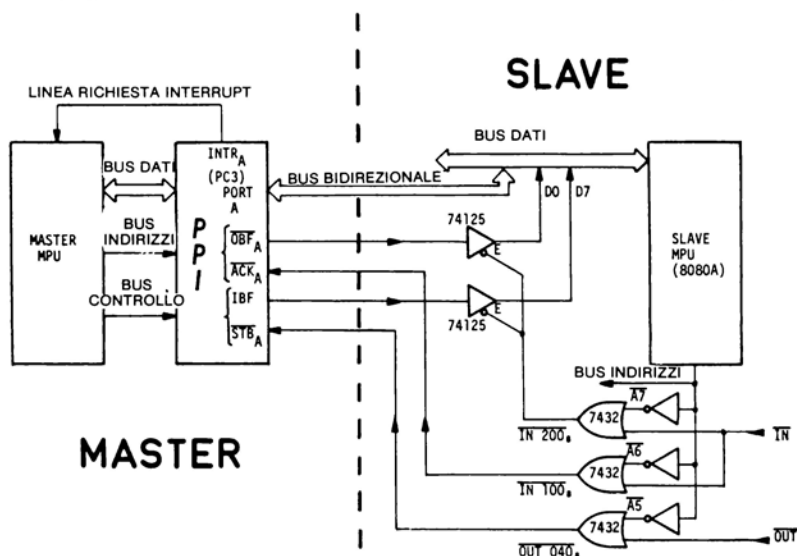



Fig. 7-6. Interfaccia tipica fra microcomputer master e slave.

all'MPU master nel solito modo, sia per I/O in mappa di memoria che in accumulatore ed è stato configurato per il funzionamento in interrupt in polling, con il collegamento della sua linea $INTR_A$ all'ingresso di interrupt della MPU master. Il microcomputer slave, è un sistema basato sull'8080A ed è collegato per il funzionamento in polling. La *caratteristica hardware* principale di un'interfaccia fra due microcomputer, è quella di assicurare che entrambi i microcomputer siano in grado di controllare i flag di handshaking per l'uscita in modo 2, $\overline{OBF_A}$ e IBF_A, del PPI.

In questo esempio l'MPU master controlla questi flag, in seguito a ricezione d'interrupt, eseguendo il polling della porta C del PPI, per una parola di stato del modo 2. L'MPU slave, è in grado di controllare questi flag tramite il collegamento delle uscite $\overline{OBF_A}$ e IBF_A del PPI al bus dati dell'MPU slave, per mezzo di un circuito integrato buffer three-state 74125. Si noti ancora che non è necessario alcun buffer three-state esterno per interfacciare la porta A al bus dati dell'MPU slave, poichè esso è presente internamente a tale porta, quando il PPI è programmato per il funzionamento in modo 2.

Un *secondo requisito hardware* dell'interfaccia master-slave, è per il microcomputer *slave*, il quale deve fornire un impulso di strobe ($\overline{STB_A}$) per il trasferimento dati da slave a master ed un impulso di accettazione ($\overline{ACK_A}$) per quelli da master a slave. In Fig. 7-6, tali impulsi sono generati dal microcomputer slave, come impulsi di selezione dispositivo rispettivamente $\overline{OUT\ 040}$ e $\overline{IN\ 100}$. Si noti che è richiesto un impulso di selezione dispositivo *d'uscita*, per pilotare la linea $\overline{STB_A}$, dal momento che i dati vengono caricati *nella* porta A del PPI e quindi sono inviati in uscita dall'MPU slave. Allo stesso modo è richiesto un impulso selezione dispositivo d'*ingresso* per pilotare la linea $\overline{ACK_A}$, dal momento che i dati vengono inviati dal PPI e quindi devono essere inseriti dall'MPU slave.

B) Una sequenza operativa

La sequenza esatta degli eventi che accadono nei trasferimenti dati fra master e slave, quando il PPI è usato come elemento d'interfaccia, dipende certamente dai compiti che devono essere assolti dall'MPU slave. In ogni caso, i trasferimenti dei dati sono governati dalle seguenti semplici regole:

(i) Nei trasferimenti dati da master a slave, il comportamento dei segnali di handshaking, $\overline{OBF}$ ed $\overline{ACK}$, è identico a quello delle operazioni d'uscita con strobe del modo 1 del PPI.

(ii) Nei trasferimenti dati da slave a master, il comportamento dei segnali di handshaking, $\overline{STB}$ ed IBF, è identico a quello delle operazioni d'ingresso con strobe del modo 1 del PPI.

(iii) La linea di richiesta interrupt della porta A del PPI, $INTR_A$, sarà posta ad uno dalla transizione allo stesso valore sia di $\overline{OBF}$ che di IBF, purchè i loro rispettivi flip-flop interni di abilitazione interrupt, INTE 1 ed INTE 2, siano al valore logico 1.

Tabella 7-1. Sequenza degli eventi illustrati nel diagramma di temporizzazione del Modo 2 di Fig. 7-7

Trasferimento Dati	Azione	Risultati
1. Dal Master al PPI	L'MPU master scrive i dati nella porta A	• Il fronte iniziale di $\overline{WR}$ azzera il flag di INTR • Il fronte finale di $\overline{WR}$ porta $\overline{OBF}$ al valore logico basso
2. Dallo slave al PPI	L'MPU slave scrive i dati nella porta A inviando un impulso di selezione dispositivo in uscita su $\overline{STB}$.	• Il fronte iniziale di $\overline{STB}$ alza IBF • Il fronte finale di $\overline{STB}$ alza il flag INTR
3. Dal PPI allo slave	L'MPU slave rileva $\overline{OBF}$ basso e legge dalla porta A, i dati provenienti dall'MPU master che sono contenuti nel buffer d'uscita della porta A. L'ingresso impulso di selezione dispositivo del microcomputer è usato come strobe di $\overline{ACK}$.	• Il fronte iniziale di $\overline{ACK}$ porta alto OBF. • Il fronte finale di $\overline{ACK}$ porta alto INTR. In questo esempio, ad ogni modo, INTR è già alto dal passo 2 di cui sopra.
4. Dal PPI al Master	Rispondendo all'interrupt di cui allo Step 2, l'MPU master rileva IBF alto e legge i dati dalla porta A, provenienti dall'MPU slave, che sono contenuti nel buffer d'ingresso della porta A.	• Il fronte iniziale di $\overline{RD}$ azzererà INTR se $\overline{OBF}$ non era alto. • Il fronte finale di $\overline{RD}$ porta basso IBF

(iv) Dal momento che la porta A, nel funzionamento in modo 2, ha un *buffer/latch d'ingresso*, per ricevere i dati provenienti dall'MPU slave ed un *buffer/latch d'uscita*, per ricevere i dati provenienti dall'MPU master, i trasferimenti dati da master a slave e viceversa, possono avvenire simultaneamente, o quasi, rispettivamente dall'MPU master e dallo slave.

Le regole sono schematicamente illustrate nel diagramma di temporizzazione di Fig. 7-7. Come esempio di una possibile sequenza operativa master-slave, guardiamo più da vicino quella illustrata in Fig. 7-7.

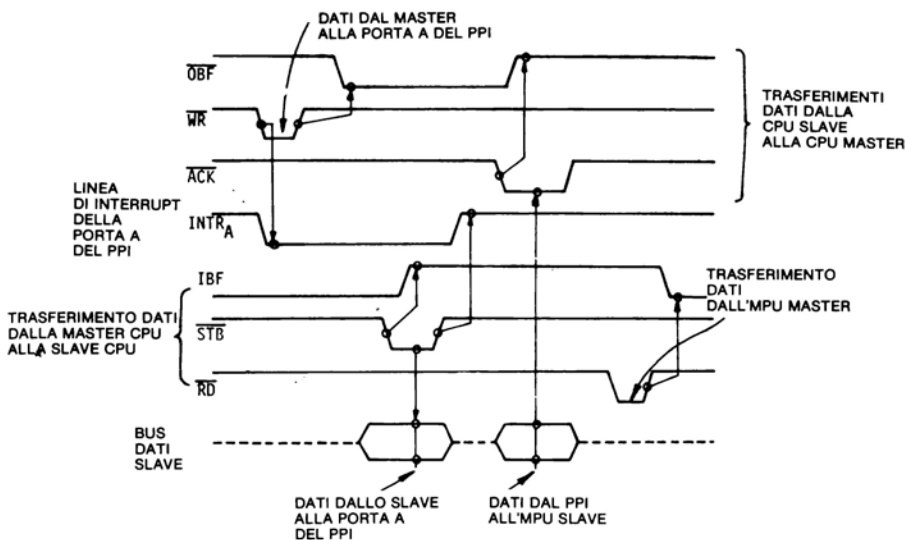



Fig. 7-7. Diagramma di temporizzazione del modo 2.

In essa è rappresentato un trasferimento dati da master a slave e viceversa. La sequenza degli eventi raffigurati in Fig. 7-7, è descritta in tabella 7-1. Si dovrà leggere tale tabella, mentre si esamina il diagramma di temporizzazione di Fig. 7-7. La caratteristica interessante di questa sequenza, è che prima che l'MPU slave abbia ricevuto il dato trasmesso per mezzo del PPI, la stessa MPU slave inizia una trasmissione di dati all'MPU master. Si noti anche il modo in cui il PPI agisce come buffer fra l'MPU master e slave ed il modo in cui agiscono i segnali di handshaking per sincronizzare i trasferimenti dati, che rappresentano ingressi asincroni per entrambe le MPU.

(C) Considerazioni software

La discussione fino ad ora, è stata concentrata sui requisiti hardware di un'interfaccia bidirezionale con il PPI in modo 2, fra microcomputer master e slave. Questo paragrafo sarebbe quindi incompleto se non si facesse alcun riferimento al software necessario per sostenere l'interfaccia. La discussione sul software è resa complessa dalla necessità di avere due programmi coordinati, interattivi, rispettivamente per il master e lo slave. Il principale dei due blocchi software, è il controllo dei flag di handshaking del PPI, $\overline{OBF}$ ed IBF. L'interfaccia master-slave di Fig. 7-6, riflette una distribuzione tipica dell'intero software di un sistema di

(A) Lettura della porta A (B) Scrittura della porta B

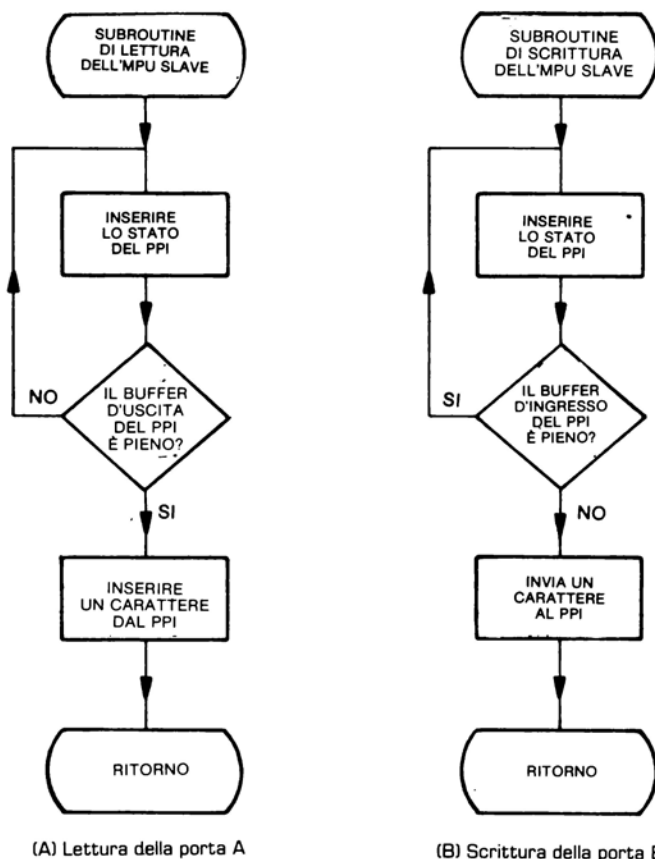



Fig. 7-8. Diagrammi di flusso delle subroutine di polling che possono essere usate dall'MPU slave in Fig. 7-6, per leggere e scrivere dati alla porta A del PPI.

controllo a microcomputer distribuito. Poichè l'MPU master avrà in generale, la responsabilità del controllo dell'intero sistema, sarà necessario che il master serva lo slave, solamente quando richiesto. Per questo motivo si adotta di solito, un sistema di interrupt per l'interfaccia MPU master e porta A del PPI. La scelta di interrupt in polling o vettorizzato, è una variabile determinata da considerazioni sul compromesso fra requisiti hardware e velocità di risposta. Dal momento che il microcomputer slave sarà dedicato a compiti particolari, la miglior velocità di risposta (trasferimento dati da master a slave) di un sistema interrupt, non è solitamente giustificata per microcomputer slave e si impiega un polling dei flag del PPI da parte dell'MPU slave, come illustrato in Fig.

7-6. Poichè il software per il microcomputer slave è il più semplice, consideriamolo per primo.

La Fig. 7-8 mostra i flowchart delle subroutine che il microcomputer slave potrà usare per leggere i dati dal PPI (trasferimento dati da slave a master). Le corrispondenti routine codificate, che si sono scritte per adattarsi alla configurazione hardware di slave di Fig. 7-6, sono riportate in Fig. 7-9.

Nel software di polling impiegato dall'MPU slave, non viene presentato alcun concetto nuovo. I flag di handshaking del modo 2, $\overline{OBF}$ ed IBF, sono collegati alle linee D0 e D7 del bus dati del microcomputer (attraverso un buffer three-state), in modo che il loro stato possa facilmente essere rilevato con uno shift a sinistra (per D7, flag IBF) ed a destra (per D0, flag $\overline{OBF}$) del contenuto dell'accumulatore nel flag di riporto (carry).

```
/
/                    SOFTWARE DELLO SLAVE
/
/(A) SUBROUTINE DI LETTURA DELLO SLAVE: TRASFERIMENTO
/DATI DA MASTER A SLAVE
/
SLAVRD, IN          /PRELEVARE I FLAG DEL PPI DAI
        200         /BUFFER THREE-STATE 74125
        RAR         /FARE SCORRERE IL FLAG OBFA DEL PPI
                    /(D0) NEL FLAG CARRY
        JNZ         /OBFA = "0"? CIOÈ I DATI SONO PRONTI
                    /PER IL
                    /TRASFERIMENTO DA MASTER A SLAVE
        SLAVRD      /NO?, PROVARE DI NUOVO
        0
        IN          /SI! INSERIRE UN CARATTERE PORTANDO
        100         /ACKA AL VALORE BASSO
        RET

/
/(B) SUBROUTINE DI SCRITTURA: TRASFERIMENTO DATI DA SLAVE
/A MASTER
/
SLAVWR, IN          /PRELEVARE I FLAG DEL PII DAI
        200         /BUFFER THREE-STATE 74125
        RAL         /FARE SCORRERE IL FLAG IBFA DEL PPI (D7)
                    /NEL FLAG CARRY
        JZ          /IBFA = "0"? CIOÈ IL PPI È PRONTO PER IL
                    /TRASFERIMENTO DA SLAVE A MASTER
        SLAVWR      /NO CONTINUARE A PROVARE
        0
        OUT         /SI BYTE O/P AL PPI AZIONANDO
        040         /STBA CON UN IMPULSO SELEZIONE
                    /DISPOSITIVO O/P
        RET
```

Fig. 7-9. Software dell'MPU slave per I/O bidirezionale fra MPU master e slave, che impiega, come interfaccia, il PPI nella configurazione del modo 2.

Il software, per il microcomputer master, è più difficile da generalizzare, a causa dei molti e svariati compiti che possono essere richiesti a questo microcomputer nel suo ruolo di supervisore e controllore dell'intero sistema. La situazione più semplice è quella in cui la porta A del PPI, quando è programmato per il funzionamento in modo 2, è l'unica sorgente di interrupt del sistema. Leggendo la porta C, per conoscere la parola di stato del modo 2 e con la mascheratura ed il rilevamento a turno, dei bit D7 ($\overline{OBF_A}$) e D5 ($\overline{IBF_A}$), l'MPU master può determinare se il PPI abbia ricevuto dei dati dal microcomputer slave ($\overline{IBF_A}$ alto), oppure se l'MPU slave abbia letto con successo i dati che il master ha inviato alla porta A ($\overline{OBF}$ alto). Nel primo caso, dove il flag di "buffer d'ingresso pieno" del PPI è alto, l'MPU master legge la porta A per avere i dati inviati dallo slave e trattenuti alla porta A. Il master elabora poi questi dati e ritorna al proprio compito principale.

Nel secondo caso, il flag $\overline{OBF_A}$, genera un interrupt ritornando ad un valore logico alto. Ciò significa che i dati, inviati al PPI dal master, sono stati letti con successo dall'MPU slave e che il buffer d'uscita della porta A è vuoto. La risposta dell'MPU master, dipende dal fatto se si deve mandare all'MPU slave, un solo byte oppure un blocco di dati. Un modo per prevedere queste due possibilità, è quello di installare un *blocco di*

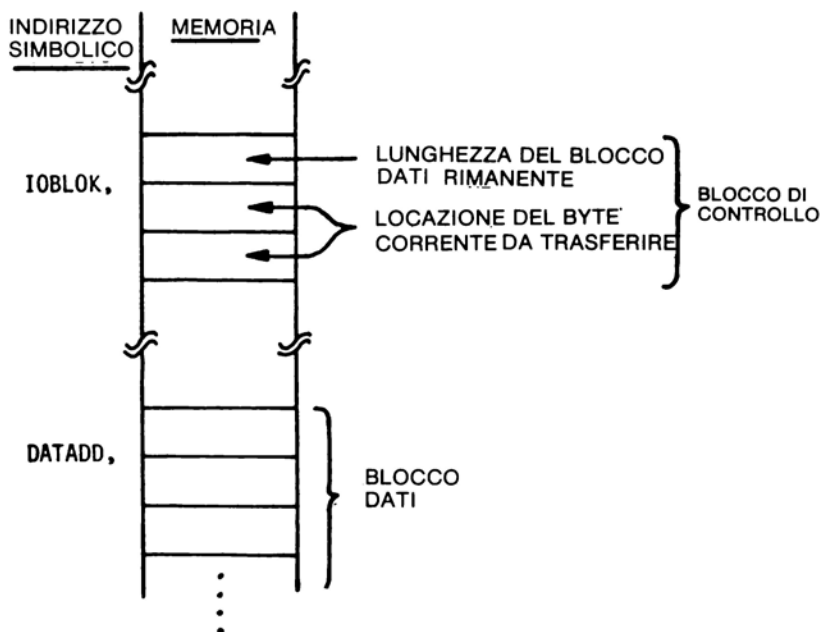



Fig. 7-10. Illustrazione schematica di un semplice blocco di controllo interrupt per un'operazione di scrittura del master nel PPI.

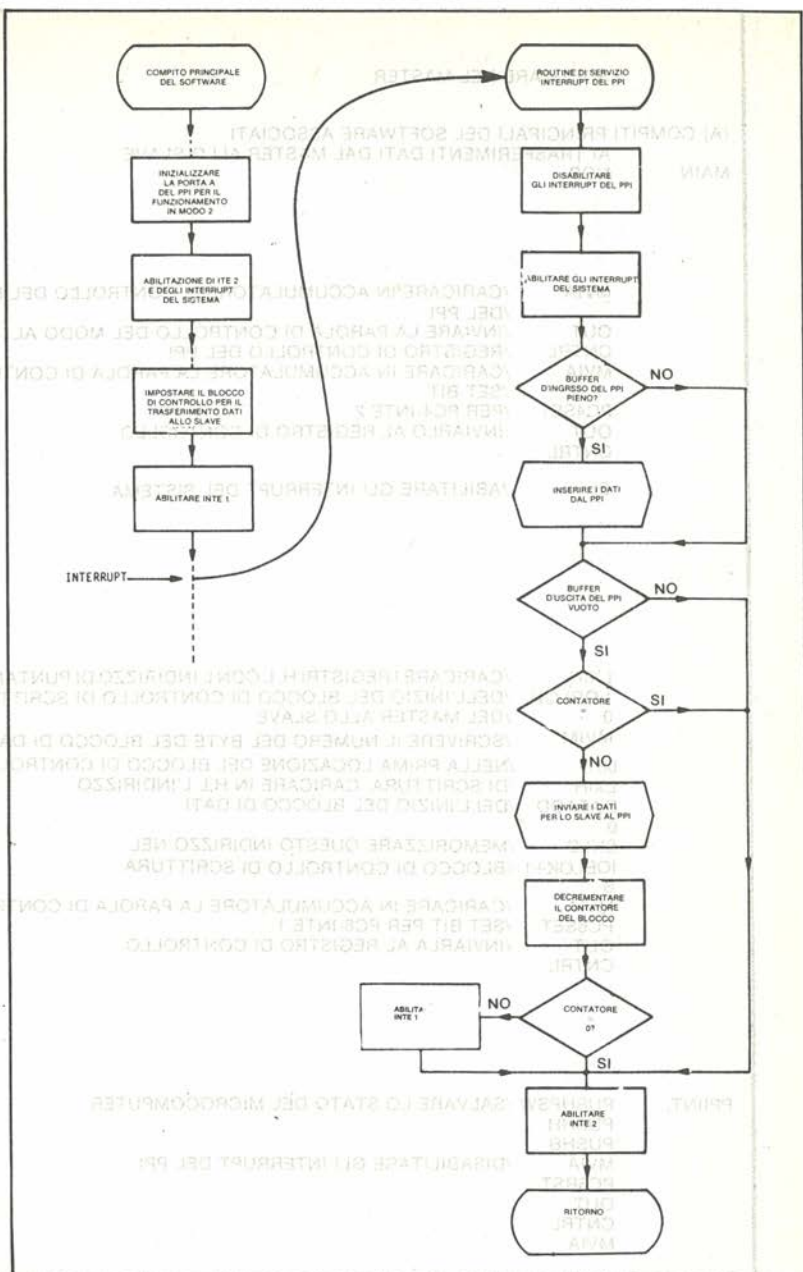



Fig. 7-11. Flowchart del software del master per i trasferimenti dei dati da master a slave.

```
            SOFTWARE DEL MASTER

(A) COMPITI PRINCIPALI DEL SOFTWARE ASSOCIATI
            AI TRASFERIMENTI DATI DAL MASTER ALLO SLAVE
MAIN,       NOP
             .
             .
             .
             .
             .
            MVIA      /CARICARE IN ACCUMULATORE IL CONTROLLO DEL MODO
                      /DEL PPI
            OUT       /INVIARE LA PAROLA DI CONTROLLO DEL MODO AL
            CNTRL     /REGISTRO DI CONTROLLO DEL PPI
            MVIA      /CARICARE IN ACCUMULATORE LA PAROLA DI CONTROLLO
                      /SET BIT
            PC4SET    /PER PC4-INTE 2
            OUT       /INVIARLO AL REGISTRO DI CONTROLLO
            CNTRL

            EI        /ABILITARE GLI INTERRUPT DEL SISTEMA
             .
             .
             .
             .
             .
             .
             .
             .
             .
            LXIH      /CARICARE I REGISTRI H, L CON L'INDIRIZZO DI PUNTAMENTO
            I OBLOK   /DELL'INIZIO DEL BLOCCO DI CONTROLLO DI SCRITTURA
            0         /DEL MASTER ALLO SLAVE
            MVIM      /SCRIVERE IL NUMERO DEL BYTE DEL BLOCCO DI DATI
            001       /NELLA PRIMA LOCAZIONE DEL BLOCCO DI CONTROLLO
            LXIH      /DI SCRITTURA. CARICARE IN H,L L'INDIRIZZO
            DATADD    /DELL'INIZIO DEL BLOCCO DI DATI
            0
            SHLD      /MEMORIZZARE QUESTO INDIRIZZO NEL
            IOBLOK+1  /BLOCCO DI CONTROLLO DI SCRITTURA
            0
            MVIA      /CARICARE IN ACCUMULATORE LA PAROLA DI CONTROLLO
            PC6SET    /SET BIT PER PC6-INTE 1
            OUT       /INVIARLA AL REGISTRO DI CONTROLLO
            CNTRL



             .
PPIINT,     PUSHPSW   /SALVARE LO STATO DEL MICROCOMPUTER
            PUSHH
            PUSHB
            MVIA      /DISABILITARE GLI INTERRUPT DEL PPI
            PC6RST
            OUT
            CNTRL
            MVIA
```

Fig. 7-12. Compito principale del software associato ai trasferimenti dati da master a slave.

```
          PC4RST
          OUT
          CNTRL
          EI        /ABILITARE GLI INTERRUPT DEL SISTEMA
          IN        /INSERIRE LA PAROLA DI STATO DEL PPI
          PORTC
          MOVBA     /SALVARE LA PAROLA DI STATO
          ANI       /MASCHERARE IBFA: - BIT D5
          040
          JZ        /IL BUFFER D'INGRESSO È PIENO (IBF = "1")
          POINTA    /NO, CONTROLLARE LO STATO DEL BUFFER D'USCITA
          0
          IN        /SI', INSERIRE UN BYTE DALLA PORTA A
          PORTA
          CALL      /ED ELABORARLO
          PROCESS
          0
POINTA,   MOVAB     /RIPRISTINARE LA PAROLA DI STATO DEL PPI
          ANI       /MASCHERARE OBFA-BIT D7
          200
          JZ        /IL BUFFER D'USCITA È VUOTO (OBF = "1")
          POINTB    /NO, RITORNO
          0
          LDA       /CONTROLLARE SE IL CONTATORE È A 0
          IOBLOK    /IN MODO CHE IL CODICE D'USCITA POSSA
          0         /ESSERE BYPASSATO SE ERA
          ORI       /L'INTERRUPT DI UN INGRESSO
          000
          JZ
          POINTB
          0
          LHLD      /CARICARE L'INDIRIZZO CORRENTE DEL BYTE DI DATI
          IOBLOK+1
          0
          MOVAM     /INSERIRE IL BYTE DI DATI DALLA MEMORIA IN UN
          OUT       /BYTE DI DATI IN USCITA ALLA PORTA A
          PORTA
          INXH      /PUNTARE L'INDIRIZZO DEL BYTE SUCCESSIVO
          SHLD      /MEMORIZZARE L'INDIRIZZO DEL PROSSIMO BYTE NEL
          IOBLOK+
          1         /BLOCCO DI CONTROLLO
          LXIH      /PUNTARE IL BYTE CONTATORE
          IOBLOK
          0
          MOVAM     /INSERIRE IL BYTE CONTATORE
          DCRA      /DECREMENTARE IL CONTATORE
          MOVMA     /SCRIVERE IL RISULTATO IN MEMORIA
          JZ        /IL CONTATORE È 0?
          POINTB    /SI', RITORNO
          0
          MVIA      /NO, CARICARE IN ACCUMULATORE IL BYTE DI CONTROLLO
          PC6SET    /PER IL SET DI PC6 = INTE 1
          OUT       /USCITA AL REGISTRO DI CONTROLLO
          CNTRL
POINTB,   MVIA      /ABILITARE INTE 2 = PC4
          PC4SET
          OUT
          CNTRL
          POPB      /RIPRISTINARE LO STATO DEL MICROCOMPUTER
          POPH
          POPPSW
          RET
```

controllo in memoria. Ciò è illustrato in Fig. 7-10 e, nel caso più semplice, sarà costituito dal numero di byte di dati che restano da trasferire all'MPU slave e dall'indirizzo, in memoria, della locazione del byte *corrente* che deve essere trasferito. Inizialmente il compito principale del software, è quello di caricare il blocco di controllo con la lunghezza del blocco di dati da trasferire allo slave, la *locazione di partenza,* in memoria, del blocco di dati e abilitare INTE 1, flip-flop di abilitazione interrupt. Tutto ciò è illustrato in Fig. 7-11, la quale riporta i flowchart per il compito principale e per la subroutine di servizio interrupt. Con INTE1 abilitato (cf Fig. 7-14) ed il buffer d'uscita del PPI vuoto ($\overline{OBF}$ alto), il PPI genererà un interrupt. Il software di interrupt, controlla lo stato del PPI e quando il buffer d'uscita della porta A viene trovato vuoto, si scrive nel PPI un byte di dati per lo slave. L'indirizzo di memoria di questo byte, si trova nel blocco di controllo. Si controlla il nuovo valore del contatore, che rappresenta il numero di byte che rimangono da trasferire, e, se non è zero, si abilita nuovamente il flag INTE 1 (sia INTE 1 che INTE 2, vengono disabilitati all'inizio della subroutine). Se il contatore è a zero, il flag INTE 1 viene lasciato disabilitato, prevenendo così ulteriori interrupt da $\overline{OBF}_A$. Quando si deve inviare un altro blocco di dati alla CPU slave, il compito principale del software sarà quello di inizializzare nuovamente il blocco di controllo con la lunghezza della stringa, con l'indirizzo di partenza dei nuovi dati e con l'abilitazione di INTE 1. I dati saranno poi trasferiti sotto il controllo di interrupt, come discritto sopra ed illustrato nel flowchart riportato in Fig. 7-11. Il software corrispondente a tali flowchart, è mostrato in Fig. 7-12.

In sistemi a microcomputer che controllano un certo numero di operazioni ingresso/uscita, che funzionano in interrupt, può essere necessario un blocco di controllo per ogni periferica. Ulteriori informazioni di controllo possono comprendere:

- Il *tipo* di I/O, cioè ingresso o uscita.
- Lo stato della corrente operazione di I/O, cioè occupato o completo.
- L'indirizzo di una subroutine che sarà chiamata dopo che il blocco I/O sia stato eseguito.
- Altri parametri associati al particolare compito di ingresso e uscita.

Ulteriori dettagli su appplicazioni tipiche di questa tecnica software, per lo smistamento di interrupt, vengono forniti nelle Note Applicative della Intel Corporation per il microcomputer SBC 80/10 ed il PPI 8255 (vedere Paragrafo 7-6).

7-5. UN'APPLICAZIONE

Nel precedente paragrafo, si sono esaminati i requisiti hardware e software per interfacciare due microcomputer, impiegando il funzionamento 2 del PPI come interfaccia. Lo scopo di questo paragrafo è quello di illustrare un ambiente tipico in cui è utile un controllo a microprocessori distribuiti e illustrare il tipo di dati che dovranno essere trasferiti fra le MPU sul bus bidirezionale. L'esempio è stato preso dall'area di controllo digitale a microcomputer di un processo industriale. Lo schema di Fig. 7-13, mostra le parti dell'intero processo. Come supervisore dell'intero processo, viene impiegato un sistema microcomputer più grande, mentre altri piccoli microcomputer dedicati, sono distribuiti nell'impianto per eseguire un controllo localizzato di piccole parti dell'intero processo. Il microcomputer principale di supervisione, dovrà probabilmente essere situato nella stanza di controllo di processo. Oltre ad una vasta memoria di lettura/scrittura e alla possibilità di floppy-disc per il programma e la memorizzazione dei dati, questo microcomputer master dovrà tipicamente controllare diverse unità video display (VDU), o terminali a raggi catodici (crt), sia all'interno della stanza di controllo che alle stazioni distribuite sull'impianto, così come il pannello di controllo del processo sulla piastra. I VDU, dovranno essere usati per l'interrogazione del sistema, per determinare lo stato in termini di grafici, istogrammi, tabulazioni, etc.

Nell'illustrazione di Fig. 7-13, il microcomputer viene impiegato per il controllo della pressione e della temperatura. I dati tipici che dovranno essere inviati al microcomputer slave dal master, comprenderanno i limiti superiori ed inferiori, così come valori da impostare per la temperatura del bagno e la pressione nelle due condotte di gas indicate. Lo slave, sarà poi responsabile del controllo della temperatura entro i limiti inviatigli dal master, così come della commutazione da una condotta di gas a un'altra, quando la pressione della linea 1, scende al di sotto del limite inferiore stabilito dal master. Nei casi in cui la pressione e la temperatura superino i limiti inviati dal master, lo slave potrà inviare un allarme o una segnalazione al master. Quando la temperatura e la pressione sono stabilizzate, si potrà inviare una segnalazione di compito assolto. Altri compiti dello slave, potranno essere la registrazione periodica e la visualizzazione della pressione e temperatura locale. Tali dati dovranno essere inviati al master periodicamente, in funzione della capacità di memoria (di solito minima) del microcomputer slave. Durante il ciclo di processo, il microcomputer master dovrà controllare i dati di temperatura e pressione provenienti dallo slave, così come l'allarme ed i flag di compito assolto e dovrà anche inviare limiti aggiornati e valori da impostare, non appena cambiano le variabili di processo in

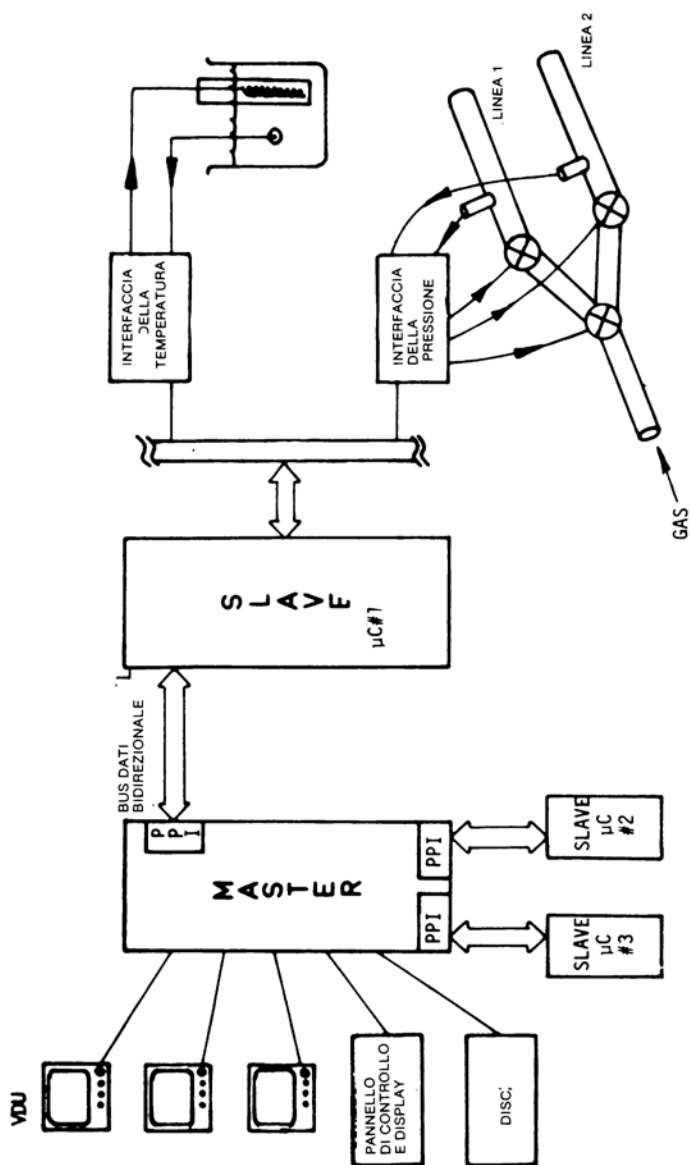



Fig. 7-13. Tipico sistema distribuito per il controllo di processo.

altre parti. Il flag di allarme, proveniente dallo slave, dovrà necessariamente essere controllato con cura dal master, in modo che, nel caso dell'insorgere di problemi, il master possa prendere provvedimenti per alterare i parametri del processo in ogni punto dell'impianto.

7-6. BIBLIOGRAFIA

Ebright, A., "8255 Programmable Peripheral Interface Applications," *Intel Application Note AP-15.*

Rolander, T., "SBC 80/10-System 80/10 Single Board Computer Applications," *Intel Application Note AP-26 (1977).*

7-7. SOMMARIO DELL'ESPERIMENTO 7-1

Esperimento	*Descrizione*
7-1	In questo esperimento, si interfacciano due microcomputer 8080 per I/O bidirezionale, impiegando il PPI nel suo funzionamento in modo 2. Viene illustrato il trasferimento sequenziale di dati dall'MPU master all'MPU slave e di nuovo indietro al master.

ESPERIMENTO 7-1
UN'INTERFACCIA BIDIREZIONALE FRA UN MICROCOMPUTER MASTER ED UNO SLAVE: FUNZIONAMENTO IN POLLING

Scopo

Gli obiettivi di questo esperimento sono i seguenti:

a. Interfacciare due microcomputer basati sull'8080 per I/O bidirezionale, impiegando un PPI programmato per funzionamento in polling in modo 2.
b. Illustrare un trasferimento sequenziale di dati dall'MPU master all'MPU slave e di nuovo indietro al master.

Discussione

In questo esperimento i microcomputer master e slave, saranno interfacciati sfruttando la tecnica descritta all'inizio di questo capitolo. Sia il microcomputer master che lo slave, controllano i segnali di handshaking di I/O bidirezionale, IBF ed $\overline{OBF}$, tramite il *polling* del loro stato. Si noti, ad ogni modo, dal circuito per l'interfaccia che il microcomputer slave, in questo esperimento, prende in ingresso IBF ed $\overline{OBF}$ rispettivamente sulle linee dati D1 e D0.

Configurazione dei pin dei circuiti integrati (Fig. 7-14)

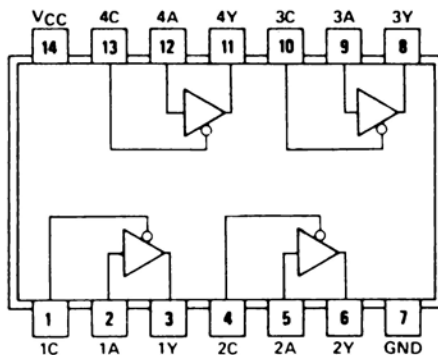
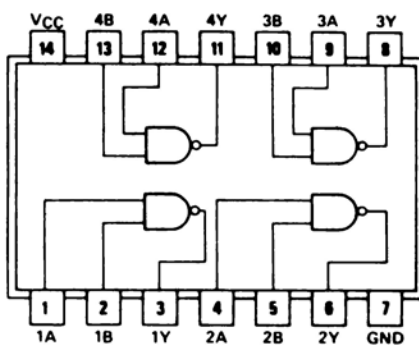
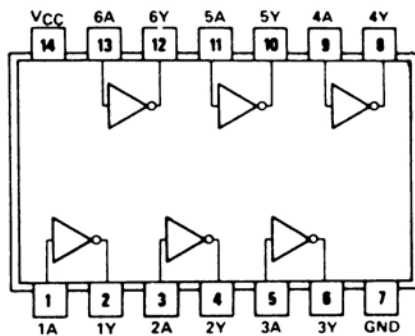



Fig. 7-14. Configurazione dei pin degli IC.

Schema del circuito (Fig. 7-15)

Fig. 7-15. Circuito per l'Esperimento 7-1.

La Fig. 7-16 mostra, sotto forma di diagramma a blocchi, il tragitto del flusso sequenziale di dati che si implementerà fra master e slave. I dati saranno trasferiti dall'MPU master al PPI, da qui al microcomputer slave, che incrementerà il byte di dati e lo ritrasmetterà al master tramite il PPI. Il master poi, incrementerà il byte ricevuto e lo ritrasmetterà allo slave. L'obiettivo è quello di programmare sia il microcomputer master che lo slave, in modo che si possa controllare facilmente questo scambio di dati.

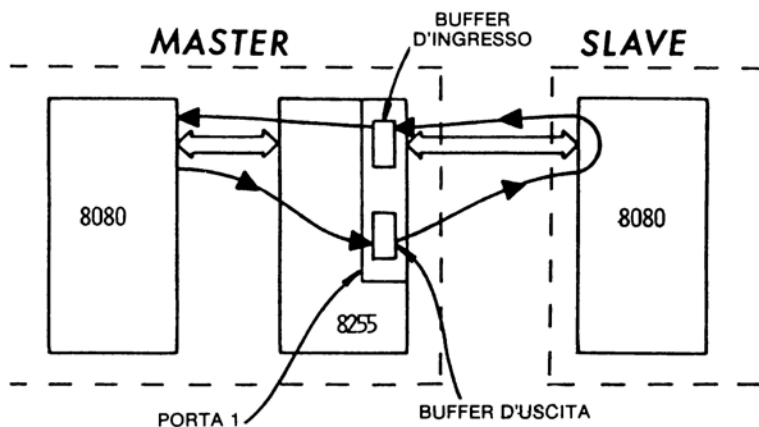



Fig. 7-16. Diagramma a blocchi del collegamento delle MPU master e slave, che mostra il flusso dei dati per l'Obiettivo b dell'Esperimento.

Per fare ciò, è necessario introdurre ritardi sia nel software del master che in quello dello slave, per far sì che trascorrano diversi secondi fra la ricezione e la trasmissione di dati sia per il master che per lo slave. Il diagramma di temporizzazione per lo scambio dei dati (Fig. 7-17), indica quattro stati stabili ed i compiti del software del master e dello slave, sono stati annotati per ognuno degli stati. Il software in Fig. 7-18 e 7-19, per il master e lo slave, implementa tali compiti.

Procedimento

Passo 1

Il primo passo è quello di collegare l'interfaccia fra i due microcomputer che devono essere usati come master e slave. Ciò può essere fatto in due passi. Innanzi tutto, interfacciare un PPI al microcomputer, designato come master, per l'ingresso/uscita in accumulatore. Molto proba-

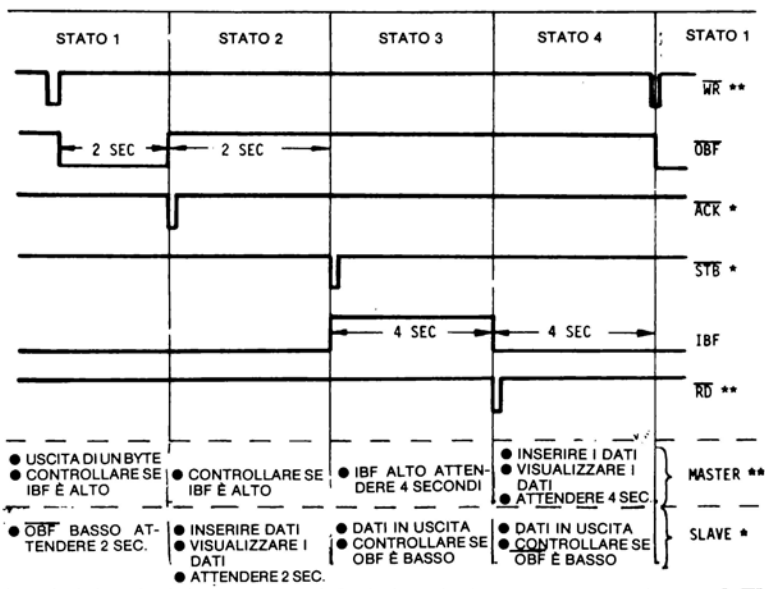



Fig. 7-17. Diagramma di temporizzazione dei segnali di handshaking bidirezionale per lo scambio di dati fra master e slave illustrato in Fig. 7-16.

bilmente, ciò sarà già stato fatto per gli esperimenti dei precedenti capitoli. Secondo, collegare le linee del PPI PA7-PA0 e PC7-PC4 al microcomputer slave, come mostrato nello schema del circuito (Fig. 7-15). Come parte di questo esercizio, si devono decodificare i seguenti impulsi di selezione dispositivo:

$$\overline{\text{IN 010}} \quad \overline{\text{OUT 040}}$$
$$\overline{\text{IN 020}} \quad \overline{\text{OUT 100}}$$

Dal momento che il microcomputer slave non richiede impulsi di selezione dispositivo aggiuntivi, si può implementare una linea d'ingresso oppure una decodifica lineare con i circuiti di Fig. 7-20. Si impieghino tali circuiti per completare l'interfacciamento del microcomputer slave al PPI. Accertarsi che ci sia un buon collegamento di massa comune fra entrambi i sistemi a microcomputer.

Passo 2

Studiare i programmi per i microcomputer master e slave, unitamente alla Fig. 7-17. In particolare, sia per il software del master che per quello dello slave, si identifichino le parti di codici corrispondenti ad ognuno

dei quattro stati individuati in Fig. 7-17. Nel software del master, si è inserito un loop senza fine nel programma, alla locazione "HERE" (020 011). Tale loop viene usato per fermare effettivamente il programma durante il test iniziale dell'interfaccia (passo 5). Esso sarà poi rimosso, per l'opportuna interazione master/slave.

Passo 3

Nel software del master, si sono usati nomi simbolici, PORT A, PORT C e CNTRL, per rappresentare rispettivamente i codici dispositivo della porta A, porta C e del registro di controllo del PPI. Tali nomi simbolici sono stati eguagliati ai codici dispositivo che sono rispettivamente 204, 206 e 207. Determinare i codici dispositivo per la porta A, porta C e registro di controllo, per interfacciare il microcomputer master al PPI. Ora, se necessario, sostituire nel programma del master i nostri codici dispositivo, con quelli giusti per la propria interfaccia PPI.

Passo 4

Con riferimento al formato della parola di controllo di modo di Fig. 2-2A, verificare che 300 sia un codice adatto per la inizializzazione del PPI in modo 2.

Passo 5

Caricare i microcomputer master e slave con i rispettivi programmi forniti. Eseguire il programma memorizzato nel microcomputer master. Che cosa si vede sulle lampade d'indicazione? Che cosa sta facendo il programma del master? Siete in grado di vedere perchè si è inserito questo passo? Annotare le risposte nello spazio sotto.

Quando il software del master veniva eseguito, si è visto che i flag $\overline{OBF}$ ed IBF, avevano entrambi assunto lo stato corrispondente al valore logico 0. Il programma è stato effettivamente bloccato alla locazione HERE, a causa del loop senza uscita. Ciò ha permesso di controllare i flag di handshaking per l'I/O bidirezionale. In questo esempio è necessario conoscere i valori iniziali di $\overline{OBF}$ ed IBF, per assicurare un trasferimento sequenziale di dati ordinato, come descritto in Fig. 7-17. In particolare se IBF, all'inizializzazione, è al valore logico 1, lo stato 2 del

diagramma sarà omesso completamente. Ciò non invaliderà la sequenza di trasferimento dei dati, ma renderà più difficile l'osservazione dei flag. Se nel vostro PPI, IBF è inizializzato al valore logico 1, si sostituisca l'istruzione di salto posta alla locazione HERE

Programmi

(A) Software del Master (Fig. 7-18)

```
                /
                /SOFTWARE DEL MASTER: POLLING.
                /
                DW STACK 024 000
                DW TIMOUT 000 277
                DB PORTA 204
                DB PORTC 206
                DB CNTRL 207
                DB MODE 300
                *020 000
020 000 061     MSTART, LXISP
020 001 000             STACK
020 002 024             0
020 003 076             MVIA     /INIZIALIZZARE IL PPI PER IL
                                 /FUNZIONAMENTC IN MODO 2
020 004 300             MODE
020 005 323             OUT
020 006 207             CNTRL
020 007 006             MVIB
020 010 000             000
020 011 303     HERE,   JMP
020 012 011             HERE
020 013 020             0
020 014 170     LOOP1,  MOVAB    /RIPRISTINARE IL BYTE
020 015 074             INRA
020 016 323             OUT      /ED INVIARLO IN USCITA ALLA PORTA 2
020 017 204             PORTA
020 020 333     LOOP2,  IN       /PRELEVARE LO STATO DEL PPI
020 021 206             PORTC
020 022 346             ANI      /CONTROLLARE IL FLAG DEL BUFFER
                                 /D'INGRESSO (PC5)
020 023 040             040
020 024 312             JZ       /IL BUFFER D'INGRESSO DELLA PORTA A È
                                 /PIENO?
020 025 020             LOOP2    /NO, PROVARE DI NUOVO
020 026 020             0
020 027 315             CALL     /SI', ATTENDERE 4 SECONDI
020 030 045             FORSEC
020 031 020             0
020 032 333             IN       /INSERIRE ORA IL BYTE DALLO SLAVE
020 033 204             PORTA
020 034 107             MOVBA    /MEMORIZZARE IL BYTE
020 035 323             OUT      /E VISUALIZZARLO
020 036 000             000
020 037 315             CALL
020 040 045             FORSEC
020 041 020             0
020 042 303             JMP      /SALTARE ALLA TRASMISSIONE DEL BYTE
                                 /ALLO SLAVE
020 043 014             LOOP1
020 044 020             0
```

Fig. 7-18. Software del master per l'Esperimento 7-1 (segue).

```
                    /
                    /SUBROUTINE DELAY "FORSEC"
                    /DESCRIZIONE: QUESTA È UNA SUBROUTINE CHE GENERA
                    /             UN RITARDO DI QUATTRO (4) SECONDI. TUTTI I
                    /             REGISTRI SONO SALVATI.
                    /
020 045 365         FORSEC, PUSHPSW/SALVARE LO STATO DEL MASTER
020 046 345                 PUSHH
020 047 305                 PUSHB
020 050 325                 PUSHD
020 051 001                 LXIB      /CARICARE LA PAROLA DI
                                      /TEMPORIZZAZIONE
020 052 310                 310       /PER UN RITARDO DI 4 SECONDI
020 053 001                 001
020 054 315         LOOPA,  CALL
020 055 277                 TIMOUT
020 056 000                 0
020 057 013                 DCXB
020 060 170                 MOVAB
020 061 261                 ORAC
020 062 302                 JNZ
020 063 054                 LOOPA
020 064 020                 0
020 065 321                 POPD      /RIPRISTINARE LO STATO DEL MASTER
020 066 301                 POPB
020 067 341                 POPH
020 070 361                 POPPSW
020 071 311                 RET
```

Fig. 7-18. Software del master per l'Esperimento 7-1.

con:

203 IN
204 PORTA

Ciò azzererà il flag IBF. Altrimenti si sostituisca il codice alle locazioni 011, 012 e 013 con NOP, cioè 000.

Passo 6

Eseguire un reset del microcomputer master ed iniziare l'esecuzione del programma. Cosa si vede questa volta?

Si dovrebbe vedere che $\overline{OBF}$ è stato posto al valore logico 0, per indicare che il software del master, ha caricato nel PPI un byte per il microcomputer slave. Il master è nello stato 1 (Fig. 7-17).

Passo 7

Si esegua ora il programma del microcomputer slave. Cosa si osserva? Ciò è coerente con il diagramma di temporizzazione di Fig. 7-17? Annotare di seguito le osservazioni.

Se si sta usando un microcomputer del tipo MMD - 1, sia come master che come slave, si vedrà il byte alla porta 0 del master, incrementato di due ogni volta che IBF andrà al valore logico 1 a 0. Ciò dovrà avvenire, circa una volta ogni 12 secondi. Allo stesso modo, il display ottale, che è guidato dal microcomputer slave, incrementerà, con riferimento al valore visualizzato alla porta 0 del master, ogni volta che $\overline{OBF}$ andrà dal valore logico 0 ad 1. Ciò dovrà avvenire, nuovamente, circa una volta ogni 12 secondi.
Se i display vengono incrementati come descritto e se i flag di handshaking seguono lo schema di Fig. 7-17, si sono interfacciati i due microcomputer con successo.

Passo 8

Si annotino, nello spazio di seguito, le locazioni del software del master e dello slave, in cui il byte ricevuto viene incrementato.

Nel software del master l'istruzione INRA, è alla locazione 020 015. Nel software dello slave, l'istruzione INRA è alla locazione 003 024.
Si provi ora a togliere queste istruzioni e ad eseguire il programma. Ci si assicuri che, quando si avviano i programmi, il software del master venga eseguito per primo. Con una delle istruzioni INRA rimossa, i display saranno incrementati di 1. Sostituire l'istruzione INRA con la DCRA. Si dovrebbe essere in grado di prevedere l'effetto di questa sostituzione e di confermare la previsione.

Passo 9

Come esercizio finale, scrivere nel seguente spazio, la locazione di memoria di lettura/scrittura della chiamata, nel software del master, al

(B) Software dello slave (Fig. 7-19)

```
                /
                /SOFTWARE DELLO SLAVE
                /
                DW STACK 004 000
                DW TIMOUT 000 277
                *003 000
003 000 061              LXISP
003 001 000              STACK
003 002 004              0
003 003 333     START,   IN        /INSERIRE I FLAG DEL PPI
003 004 010              010
003 005 346              ANI       /IL BUFFER D'USCITA DEL PPI È PIENO?
003 006 001              001
003 007 302              JNZ       /NO, PROVARE DI NUOVO
003 010 003              START
003 011 003              0
003 012 315              CALL      /SI', ATTENDERE 2 SECONDI
003 013 032              TWOSEC
003 014 003              0
003 015 333              IN        /INSERIRE I DATI DALLA PORTA A
003 016 020              020
003 017 323              OUT       /VISUALIZZARE I DATI
003 020 040              040
003 021 315              CALL
003 022 032              TWOSEC
003 023 003              0
003 024 074              INRA      /INCREMENTARE IL BYTE
003 025 323              OUT       /INVIARE IL BYTE AL MASTER
003 026 100              100
003 027 303              JMP       /SALTARE AD INSERIRE UN BYTE DAL
                                    MASTER
003 030 003              START
003 031 003              0
                /
                /
                /
                /SUBROUTINE DELAY "TWO SEC"
                /DESCRIZIONE: QUESTA È UNA SUBROUTINE CHE GENERA
                /             UN RITARDO DI  DUE (2) SECONDI. TUTTI I REGISTRI
                /             SONO SALVATI
                /
003 032 365     TWOSEC,PUSHPSW/SALVARE LO STATO DELLO SLAVE
003 033 345              PUSHH
003 034 305              PUSHB
003 035 325              PUSHD
003 036 016              MVIC      /CARICARE IL BYTE DI TEMPORIZZAZIONE
003 037 310              310       /PER UN RITARDO DI 2 SECONDI
003 040 315     LOOPA,   CALL
003 041 277              TIMOUT
003 042 000              0
003 043 015              DCRC
003 044 302              JNZ
003 045 040              LOOPA
003 046 003              0
003 047 321              POPD      /RIPRISTINARE LO STATO DELLO SLAVE
003 050 301              POPB
003 051 341              POPH
003 052 361              POPPSW
003 053 311              RET
```

Fig. 7-19. Software dello slave per l'Esperimento di Fig. 7-1.

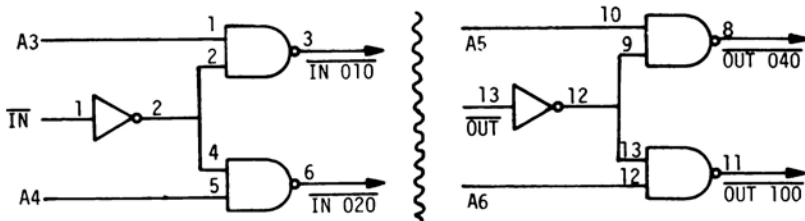



Fig. 7-20. Circuiti per una sola linea indirizzo o decodifica lineare.

ritardo di 4 secondi, il quale genera lo stato 4; scrivere, inoltre, la locazione di chiamata, nel software dello slave, del ritardo di 2 secondi il quale genera lo stato 2.

La chiamata per lo stato 4, nel software del master, è alla locazione 020 037. Per lo stato 2, nel software dello slave, la chiamata è alla locazione 003 021. Sostituire tali chiamate con NOP ed eseguire di nuovo il software del master e dello slave, iniziando da quello del master. Si sono rimossi gli stato 2 e 4 ? Dovrebbe essere così.

Domande

1. Tracciare un diagramma di temporizzazione, simile a quello di Fig. 7-17, per la seguente sequenza di quattro stati:

   (a) Il master invia in uscita un byte al PPI.
   (b) Lo slave invia in uscita un byte al PPI.
   (c) Il master preleva un byte dal PPI.
   (d) Lo slave preleva un byte dal PPI.

2. Modificare il software del master e dello slave, per generare questa sequenza di trasferimento dati.

APPENDICE 1

CARATTERISTICHE ELETTRICHE E DIAGRAMMI DI TEMPORIZZAZIONE DELL'8255*

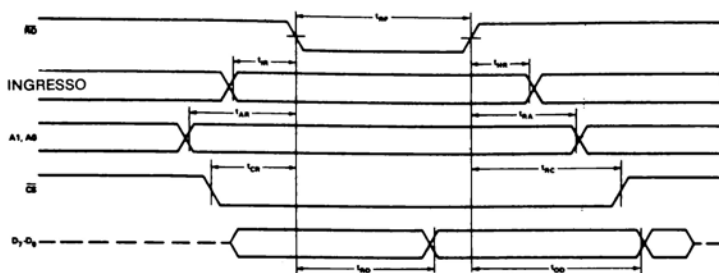
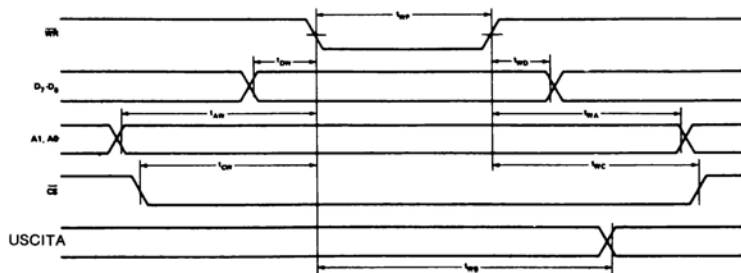
* *Per gentile concessione della Intel Corporation*

CARATTERISTICHE C.C. T_A = 0°C fino a 70°C; V_{CC} = +5V ±5%; V_{SS} = 0V

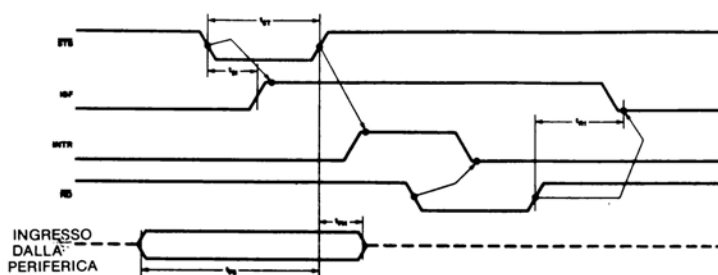
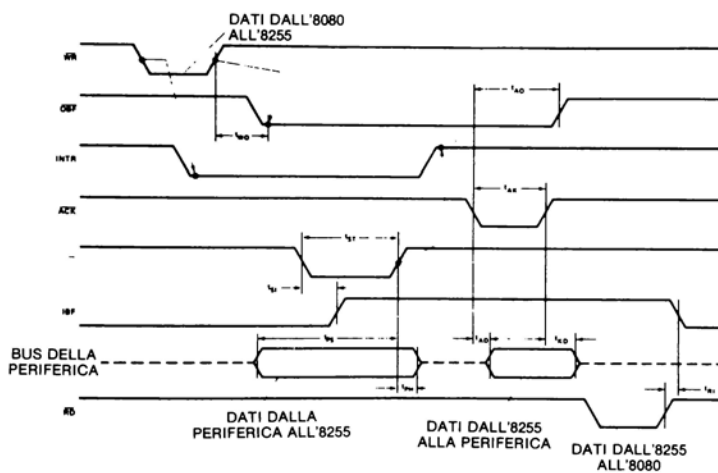
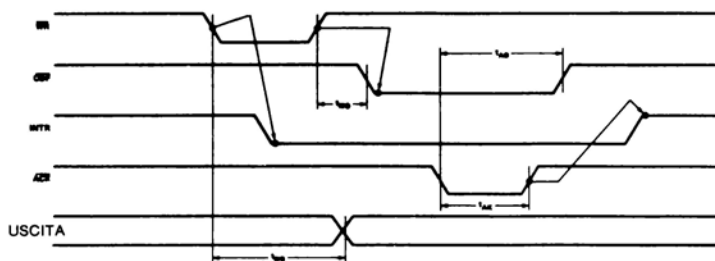
Simbolo	Parametro	Min.	Tip.	Max.	Unità	Condizioni di test
V_{IL}	Input Low Voltage			.8	V	
V_{IH}	Input High Voltage	2.0			V	
V_{OL}	Output Low Voltage			.4	V	I_{OL} = 1.6mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -50µA (-100µA per la porta D.B.
I_{OH}[1]	Darlington Drive Current		2.0		mA	V_{OH} = 1.5V, R_{EXT} = 390Ω
I_{CC}	Power Supply Current		40		mA	

NOTA:
1. Disponibile solo su 8 pin

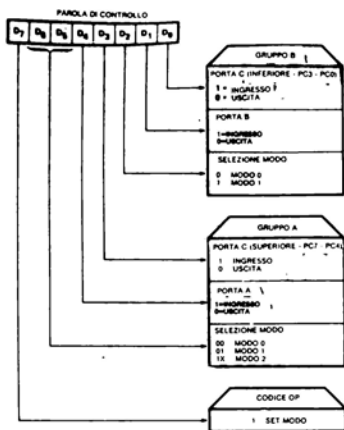
CARATTERISTICHE C.A. T_A = 0°C fino a to 70°C; V_{CC} = +5V ±5%; V_{SS} = 0V

Simbolo	Parametro	Min.	Tip.	Max.	Unità	Condizioni di test
t_{WP}	Pulse Width of $\overline{WR}$			430	ns	
t_{DW}	Time D.B. Stable Before $\overline{WR}$	10			ns	
t_{WD}	Time D.B. Stable After $\overline{WR}$	65			ns	
t_{AW}	Time Address Stable Before $\overline{WR}$	20			ns	
t_{WA}	Time Address Stable After $\overline{WR}$	35			ns	
t_{CW}	Time CS Stable Before $\overline{WR}$	20			ns	
t_{WC}	Time CS Stable After $\overline{WR}$	35			ns	
t_{WB}	Delay From $\overline{WR}$ To Output			500	ns	
t_{RP}	Pulse Width of $\overline{RD}$	430			ns	
t_{IR}	$\overline{RD}$ Set-Up Time	50			ns	
t_{HR}	Input Hold Time	50			ns	
t_{RD}	Delay From $\overline{RD}$ = 0 To System Bus	350			ns	
t_{DD}	Delay From $\overline{RD}$ = 1 To System Bus	150			ns	
t_{AR}	Time Address Stable Before $\overline{RD}$	50			ns	
t_{CR}	Time $\overline{CS}$ Stable Before $\overline{RD}$	50			ns	
t_{AK}	Width Of $\overline{ACK}$ Pulse	500			ns	
t_{ST}	Width Of $\overline{STB}$ Pulse	350			ns	
t_{PS}	Set-Up Time For Peripheral	150			ns	
t_{PH}	Hold Time For Peripheral	150			ns	
t_{RA}	Hold Time for A_1, A_0 After $\overline{RD}$ = 1	379			ns	
t_{RC}	Hold Time For CS After $\overline{RD}$ = 1	5			ns	
t_{AD}	Time From $\overline{ACK}$ = 0 To Output (Mode 2)			500	ns	
t_{KD}	Time From $\overline{ACK}$ = 1 To Output Floating			300	ns	
t_{WO}	Time From $\overline{WR}$ = 1 To $\overline{OBF}$ = 0			300	ns	
t_{AO}	Time From $\overline{ACK}$ = 0 To $\overline{OBF}$ = 1			500	ns	
t_{SI}	Time From $\overline{STB}$ = 0 To IBF			600	ns	
t_{RI}	Time From $\overline{RD}$ = 1 To IBF = 0			300	ns	

Modo 0 (Ingreso Base)

Modo 0 (Uscita Base)

Modo 1 (Ingresso con Strobe)

Modo 1 (Uscita con Strobe)

Modo 2 (Ingresso/Uscita bidirezionale)

APPENDICE 2

SOMMARIO DELLE PAROLE DI CONTROLLO E DI STATO DELL'8255*

* *Per gentile concessione della Intel Corporation*

PAROLA DI CONTROLLO
D7 D6 D5 D4 D3 D2 D1 D0
GRUPPO B
PORTA C (INFERIORE - PC3 - PC0)
1 = INGRESSO
0 = USCITA
PORTA B
1=INGRESSO
0=USCITA
SELEZIONE MODO
0 MODO 0
1 MODO 1
GRUPPO A
PORTA C (SUPERIORE - PC7 - PC4)
1 INGRESSO
0 USCITA
PORTA A
1=INGRESSO
0=USCITA
SELEZIONE MODO
00 MODO 0
01 MODO 1
1X MODO 2
CODICE OP
1 SET MODO
PAROLA DI CONTROLLO DEL MODO
BIT PORTA C
STATO DEL GRUPPO A
STATO DEL GRUPPO B
INGRESSO USCITA MODO 0
PORTA D INGRESSO MODO 1
PORTA D USCITA MODO 1
PAROLA DI STATO DEL MODO 2
PAROLA DI CONTROLLO
FLAG SET RESET
0 RESET BIT
1 SET BIT
SELEZIONE DEL BIT
PORT C BIT
NON USATO PORRE A 000
CODICE OP
0 SET RESET BIT
PAROLA DI CONTROLLO SET/RESET BIT
BIT PORTA C
STATO DEL GRUPPO A
STATO DEL GRUPPO B
PORTA D INGRESSO
PORTA D USCITA
PORTA D'INGRESSO
PORTA D'USCITA
PAROLA DI STATO DEL MODO 1

45

INTERFACCIAMENTO DI MICROCOMPUTER

ESPERIMENTI UTILIZZANTI IL

CHIP 8255 PPI

Paul F. Goldsbrough
con esperimenti di
Peter R. Rony

GRUPPO
EDITORIALE
JACKSON